# GLAP-Box v3 : Manuel d'utilisation et test des fonctionnalités.

## *Introduction*

Une fois que vous avez monté votre GLAP-Box et que le système GLAP-Box est installé (ou lancé en live-USB), vous êtes potentiellement opérationnnel pour réaliser un système embarqué, permettant la programmation à distance de la carte Arduino embarquée, d'utiliser la synthèse vocale, le retour vidéo, le partage de fichier, la reconnaissance visuelle, le suivi d'objet, etc...

Dans ce manuel, les différentes procédures courantes d'utilisation et de test sont décrites.

**Remarque**

Tous les codes fournis dans cette documentation fonctionnent directement sur la GLAP-Box par simple copier/coller et lancement de l'exécution côté Arduino ou Processing selon les cas.

## *Accès au bureau distant par VNC*

Cette procédure est la procédure de base pour utiliser et contrôler la GLAP-Box à distance.

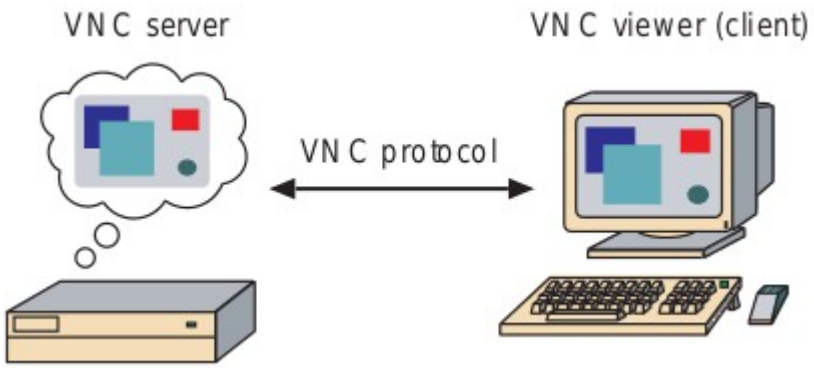
## Le matériel utilisé

Vous avez besoin :

- d'un routeur ou d'une box internet, wifi ou ethernet ou les 2
- de la GLAP-Box + câble éthernet (impératif pour une première connexion) ou d'une clé USB wifi (l'autoconnexion sera fonctionnelle uniquement après une première connexion par ethernet)
- du poste fixe permettant d'accéder à la GLAP-Box connecté sur le même réseau que celui utilisé pour la GLAP-Box.

## Principe d'accès distant au bureau de la GLAP-Box

La GLAP-Box est une unité centrale de PC classique de taille réduite, mais sans écran, ni clavier, ni souris. Pour en prendre le contrôle, on va accéder à distance, via un réseau, au bureau de la GLAP-Box et y travailler comme si on se trouvait face à la GLAP-Box. L'intérêt de procéder ainsi est de gérer la GLAP-Box à distance et même sans fil depuis son poste fixe. Ceci est très intéressant dans le cas d'un robot mobile, d'un système de mesure ou de capture vidéo d'accès difficile ou peu pratique, etc...




Principe de l'accès distant par VNC.

La GLAP-Box est configurée par défaut en serveur VNC.

Le protocole VNC :

- est opensource
- est indépendant de la plateforme, ainsi on peut accéder à la GLAP-Box depuis un MAC, Windows, Linux et même Android.
- Est peu sécurisé, ce qui est sans importance sur un réseau local dédié. Mais si on souhaite plus de sécurité, on pourra tunnelliser la connexion VNC par SSH (voir le manuel « GLAP-Box from Scratch » pour le détail de la procédure)
- Supporte le retour vidéo !

## Le réseau utilisé

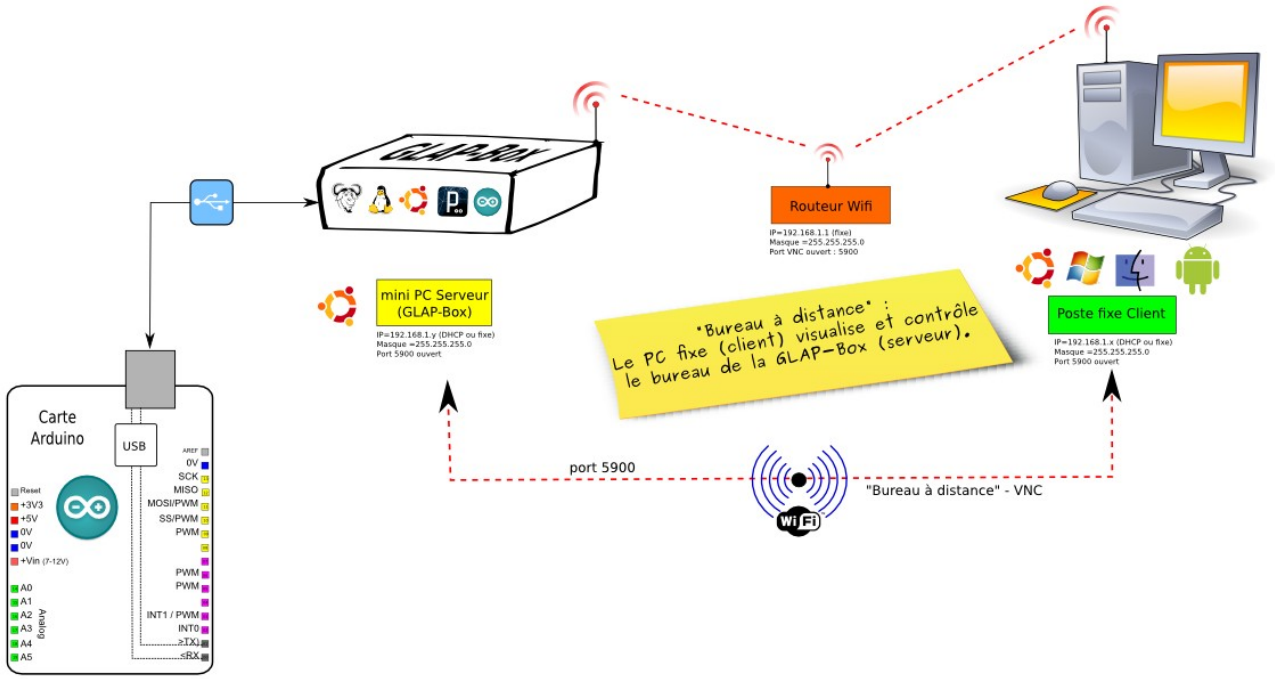
On considère ici que l'accès distant au bureau de la GLAP-Box se fait par VNC (Virtual Network Computing) sur un réseau local disposant d'un routeur. Le réseau doit donc disposer au minimum :

- du **routeur du réseau** : soit une Box internet classique, soit un routeur dédié, wifi et/ou ethernet selon le mode de connexion utilisé.
- de la **GLAP-Box** connectée au routeur soit en wifi, soit en éthernet filaire, (pour une première utilisation, utiliser une connexion ethernet filaire)
- du **poste fixe** (sous Gnu/Linux, Windows ou Mac Os X voire Android) permettant d'accéder au bureau de la GLAP-Box par VNC.

Noter que toutes les procédures d'utilisation de la GLAP-Box ont été testées et sont garanties fonctionnelles avec **un poste fixe de contrôle sous Xubuntu 12.04 LTS**. Cela devrait également fonctionner sous les autres OS (mais sans garanties...) .

**Ceci étant, remarquer que quelque soit votre OS sur le poste fixe, vous pouvez utiliser toutes les fonctions avancées de la GLAP-Box sans rien modifier sur votre système !**

**Une bonne façon de tester Gnu/Linux, Arduino, Processing, OpenCV, etc... !**




**IMPORTANT**

Les ports **22**(ssh), **111** (rpcbind), **2049** (nfs), **5800** (vnc-http) et **5900** (vnc) doivent être ouverts sur le routeur utilisé.

## Les adresses IP sur le réseau local utilisé

On considère ici que l'attribution des adresses sur le réseau se fait automatiquement par DHCP. La GLAP-Box est configurée par défaut pour fonctionner en DHCP.

Typiquement, sur un réseau avec une box internet, les adresses IP des différents postes seront :

- **192.168.1.1** pour le routeur,
- **192.168.1.x** pour le poste fixe, (attribuée automatiquement par le routeur DHCP)
- **192.168.1.y** pour la GLAP-Box, (attribuée automatiquement par le routeur DHCP)

où les 3 premiers groupes de chiffres représentent le numéro du réseau (**192.168.1**)et les chiffres **1, x** et **y** les numéros de chaque poste sur le réseau, le routeur ayant typiquement toujours le n°1. Les numéros x et y sont attribués automatiquement par le routeur DHCP aux autres postes au moment de leur connexion. Pour être complet, on en déduit que le masque du réseau est **255.255.255.0.**

**Truc à connaître :**

Si on utilise à la fois des IP fixes et des IP attribuées automatiquement par DHCP sur le même réseau (si si, on peut !), cela est possible si les adresses IP fixes utilisées sont en dehors de la plage d'adresses utilisée par le routeur DHCP. A noter qu'il est également possible sur certains routeurs ou box d'attribuer une adresse IP précise à un matériel donné (défini par son adresse MAC). Pour savoir quelle plage d'adresse votre routeur utilise, s'y connecter depuis un navigateur avec l'adresse **192.168.1.1**

# Découvrir les adresses IP utilisées sur le réseau local.

Une fois que le réseau est matériellement constitué et que tous les éléments sont sous tension, il devient possible de connaître les adresses IP de l'ensemble des postes du réseau. Cette procédure peut sembler « lourde » la première fois, mais on s'y habitue vite et ce n'est pas si sorcier que çà !

## *Déterminer l'adresse IP du poste fixe :*

Tout d'abord, sur le poste fixe, il faut obtenir l'adresse IP qui lui a été attribuée automatiquement par le routeur DHCP. Sur un poste Gnu/Linux, ouvrir un Terminal et saisir la ligne de commande :

```
$ ifconfig
```

On obtient quelque chose de la forme :

```
eth0      Link encap:Ethernet  HWaddr xx:xx:xx:xx:xx:xx
          inet adr:192.168.1.10  Bcast:192.168.1.255  Masque:255.255.255.0
          adr inet6: fe80::221:85ff:fe3f:705b/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Packets reçus:109 erreurs:0 :0 overruns:0 frame:0
          TX packets:81 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:15129 (15.1 KB) Octets transmis:16402 (16.4 KB)
          Interruption:43 Adresse de base:0xc000

lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          Packets reçus:2071 erreurs:0 :0 overruns:0 frame:0
          TX packets:2071 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          Octets reçus:91741 (91.7 KB) Octets transmis:91741 (91.7 KB)

ra0       Link encap:Ethernet  HWaddr xx:xx:xx:xx:xx:xx
          inet adr:192.168.3.106  Bcast:192.168.3.255  Masque:255.255.255.0
          adr inet6: fe80::9afc:11ff:febf:24c1/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Packets reçus:64118 erreurs:0 :0 overruns:0 frame:0
          TX packets:14347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:22328447 (22.3 MB) Octets transmis:1252998 (1.2 MB)
```

Sur ce poste, il y a 3 interfaces réseaux identifiées chacune par les lettres de la première colonne :

- l'interface eth0 correspondant à un réseau éthernet
- l'interface lo correspondant au réseau local sur le poste fixe lui-même,
- l'interface ra0 correspondant à un réseau wifi

L'adresse IP de chaque interface est de la forme xxx.xxx.xxx.xxx et est précédée de « inet adr : », qui ici nous donne :

- 192.168.1.10 pour l'interface réseau éthernet eth0
- 127.0.0.1 pour l'interface lo du réseau local du poste fixe
- 192.168.3.106 pour l'interface wifi ra0

Par ailleurs, remarquer que le masque réseau est bien de la forme 255.255.255.0 pour les 3 interfaces, donc seul le dernier chiffre indique le numéro du poste fixe sur chaque réseau.

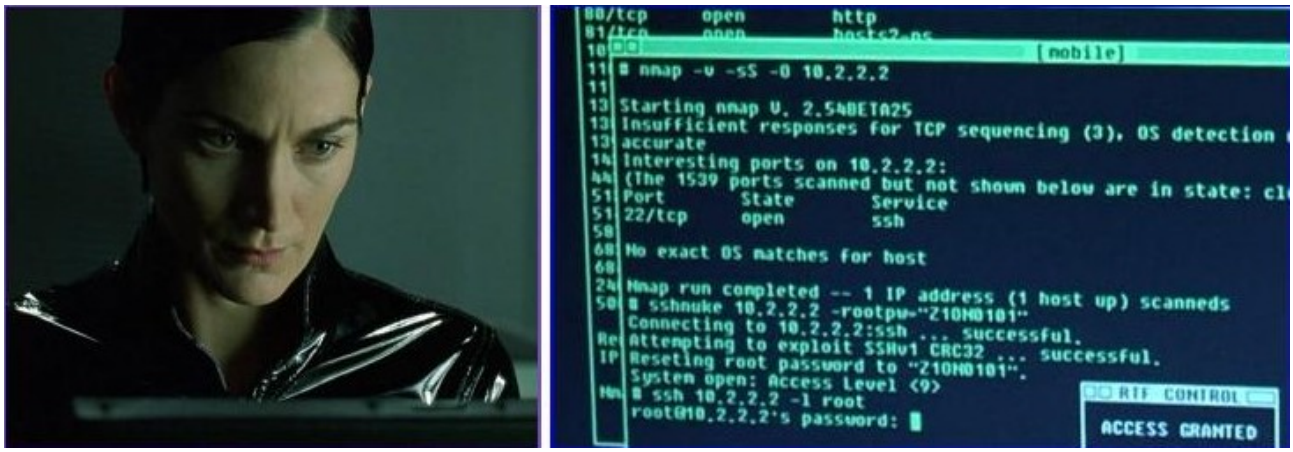
Ainsi, le poste fixe est identifié avec :

- le numéro 10 sur le réseau 192.168.1. auquel est connecté l'interface eth0

- le numéro  106 sur le réseau 192.168.3. auquel est connecté l'interface ra0

## *Scanner le réseau local et obtenir l'adresse IP de la GLAP-Box*

Une fois que l'on connaît le numéro du réseau et l'adresse du poste fixe, pour connaître l'adresse de la GLAP-Box, le plus simple est de « scanner » le réseau à partir du poste fixe. Ceci se fait sous Xubuntu à l'aide d'un petit logiciel appelé nmap qui existe aussi en version graphique, appelé Zenmap. Pour la petite histoire, c'est ce logiciel qui est utilisé dans Matrix et autres films de « pirates informatiques », mais rassurez-vous, vous ne faites rien d'illégal en l'utilisant !


80/tcp open http
[mobile]
# nmap -v -sS -O 10.2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection
accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: cl
Port State Service
22/tcp open ssh
No exact OS matches for host
Nmap run completed -- 1 IP address (1 host up) scanneds
# sshnuke 10.2.2.2 -rootpw="Z10N0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z10N0101".
System open: Access Level (9)
# ssh 10.2.2.2 -l root
root@10.2.2.2's password:
RTF CONTROL
ACCESS GRANTED

Donc, commencer par installer nmap, soit par Synaptic, soit en ligne de commande :

```
$ sudo apt-get install nmap
```

Une fois fait, on lance un « scan » du réseau où est connecté la GLAP-Box. Ici, nous considérons que la GLAP-Box est connectée sur le réseau **192.168.1**. à une adresse que nous cherchons à connaître. Afin de chercher tous les postes (numéro entre 1 et 255), et donc la GLAP-Box, connecté sur le réseau 192.168.1., on va donc saisir simplement la ligne de commande :

```
$ sudo nmap 192.168.1.1-255
```

On obtient quelque chose comme çà :

```
Starting Nmap 5.21 ( http://nmap.org ) at 2012-07-20 12:38 CEST
Nmap scan report for HSIB.home (192.168.1.1)
Host is up (0.016s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
23/tcp   open  telnet
80/tcp   open  http
443/tcp  open  https
992/tcp  open  telnets
8443/tcp open  https-alt
MAC Address: xx:xx:xx:xx:xx:xx

Nmap scan report for xavier.home (192.168.1.10)
Host is up (0.0000070s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
80/tcp   open  http
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap scan report for new-host-3.home (192.168.1.12)
Host is up (0.00021s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs
```

```
5800/tcp open  vnc-http
5900/tcp open  vnc
MAC Address: xx:xx:xx:xx:xx:xx
```

Ici, on retrouve :

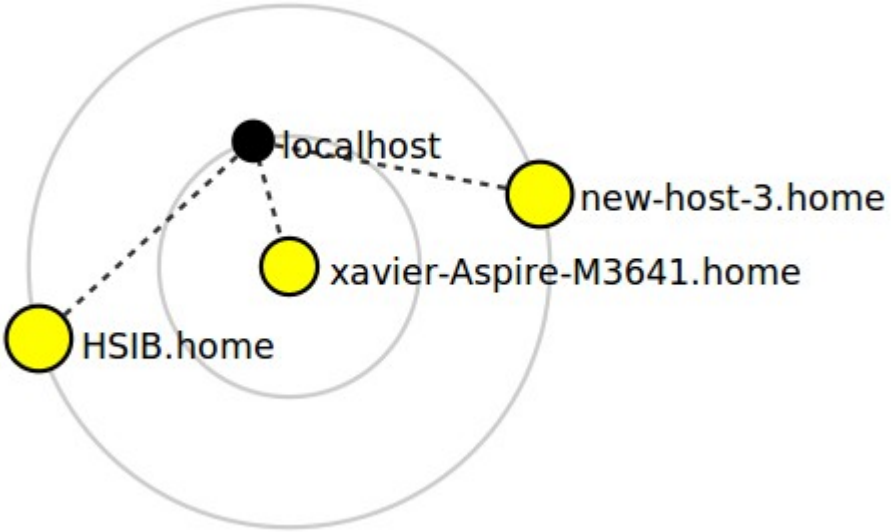
- le routeur avec l'adresse **192.168.1.1**
- le poste fixe avec l'adresse déjà obtenue précédemment : **192.168.1.10**
- et le 3ème poste qui est ici la GLAP-Box avec l'adresse : **192.168.1.12**

Si il y a « du monde » sur le réseau local, faire un premier « scan » avec la GLAP-Box éteinte puis un second « scan » avec la GLAP-Box allumée : le poste en plus sur le réseau est la GLAP-Box.

Après une première connexion, en notant l'adresse MAC de la GLAP-Box, on pourra facilement la reconnaître lors des connexions suivantes. On peut la reconnaître également à l'aide des ports ouverts...

Noter que nmap fournit également les ports ouverts pour chaque poste : on voit ici que sur la GLAP-Box, les ports 22 (sécurisation ssh), 111 et 2049 (partage de fichier nfs), 5800 et 5900 (accès vnc) sont ouverts sur la GLAP-Box.

Pour info, Zenmap, l'implémentation graphique de nmap, fournit une visualisation graphique du réseau, ce qui est aussi sympa (onglet topologie) :




## *Se connecter par VNC à la GLAP-Box*

Une fois que l'on dispose de l'adresse IP de la GLAP-Box, les choses sont simples. Il suffit de lancer le client VNC sur le poste fixe et de se connecter à l'adresse IP de la GLAP-Box.

Côté GLAP-Box, il n'y a rien à faire : elle est déjà paramétrée en serveur VNC si on utilise l'image ISO du système. Pour le détail de l'activation du serveur VNC sur la GLAP-Box, voir au besoin le document « GLAP-Box from scratch ».

Sous Xubuntu, un client VNC léger et rapide est xtightvncviewer. Il y a d'autres alternatives, notamment graphiques, mais elles sont plus lentes à mon goût.

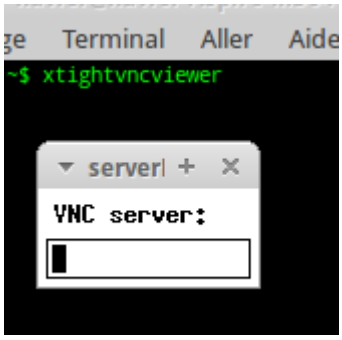
On commence par installer sur le poste fixe le programme xtightvncviewer si ce n'est déjà fait, soit avec Synaptic, soit en ligne de commande :

```
$ sudo apt-get install xtightvncviewer
```

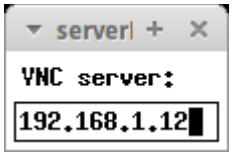
Ensuite, en ligne de commande, on lance le programme avec la commande :

```
$ xtightvncviewer
```

On obtient alors une petite fenêtre où il faut saisir l'adresse de la GLAP-Box :

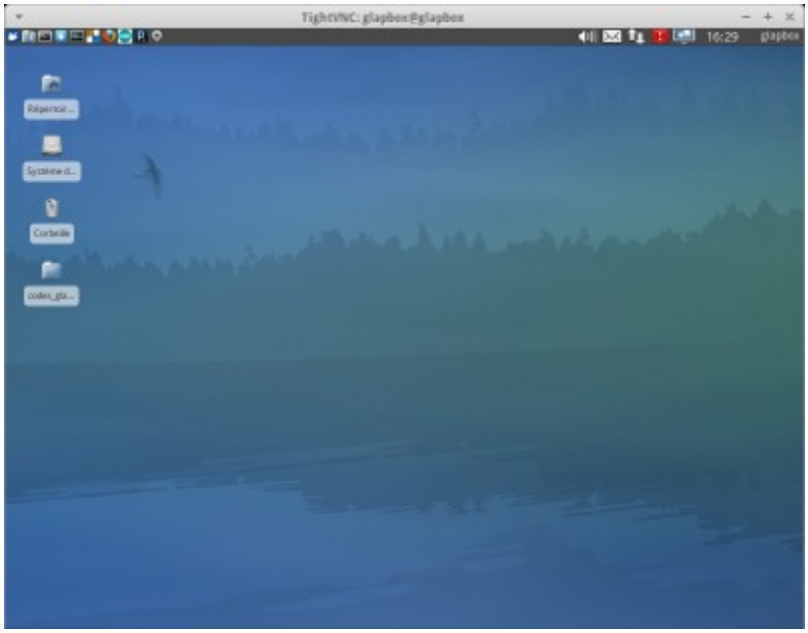



Saisir alors l'adresse IP de la GLAP-Box et valider :

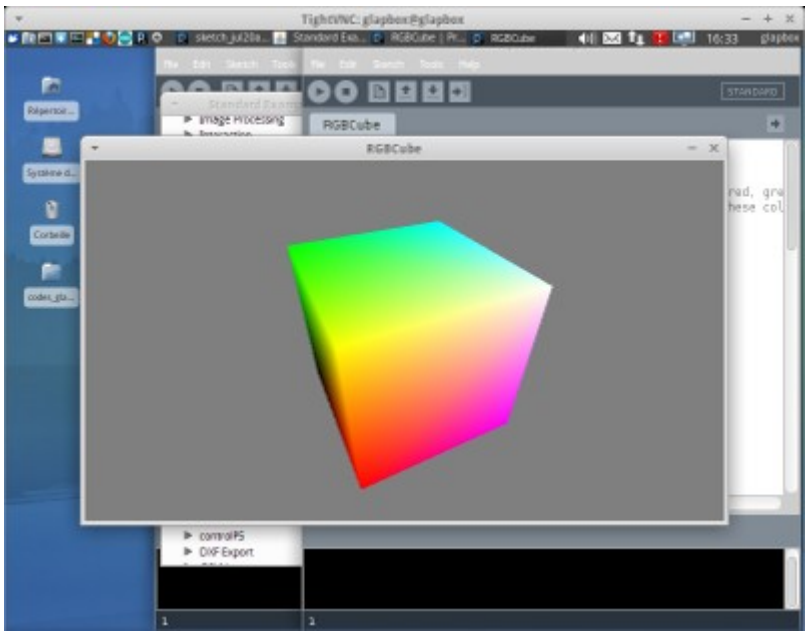



On obtient alors une seconde fenêtre (çà peut prendre quelques secondes, notamment par wifi..) où il faut saisir un mot de passe (glapbox par défaut) puis valider. Noter qu'il est possible de modifier ce mot de passe voire même de le supprimer.

Et là, MAGIE ! Vous devez voir après quelques instants le bureau de votre GLAP-Box : çà y est, vous pouvez travailler sur la GLAP-Box comme si vous étiez sur votre PC !



A titre de simple test, lancer Processing (lanceur dans le menu du haut) et lancer l'exemple 3D RGBCube :




Votre GLAP-Box peut se trouver à l'autre bout de la pièce ou de la maison ou encore sur votre robot mobile, vous travaillez dessus sans y toucher ! Vous allez ainsi pouvoir reprogrammer votre carte Arduino de façon distante, récupérer le retour vidéo, etc...

Truc à connaître : certains routeurs et box internet en mode DHCP permettent malgré tout d'attribuer une adresse IP pré-définie à un élément du réseau en connaissant l'adresse MAC du matériel en question : si on veut se simplifier la vie, on pourra réserver une telle adresse pour sa GLAP-Box sur le routeur utilisé.

## *Stopper la connexion VNC*

Pour mettre fin à la connexion VNC, c'est simple : il suffit de fermer la fenêtre VNC ! Par contre, votre GLAP-Box n'est pas éteinte pour autant et continue de fonctionner dans l'état où vous l'avez quittée. Vous pourrez dès lors vous reconnecter de la même façon à tout moment.

Pour éteindre également la GLAP-Box, le faire avant de se déconnecter ou appuyer brièvement sur le bouton ON/OFF de la GLAP-Box : l'appui déclenchera l'extinction.

## *Améliorer la fluidité de la connexion VNC*

Pour améliorer la fluidité de la connexion VNC, il est possible très simplement de le faire avec **xtightvncviewer** à l'aide de la commande suivante qui va fixer le taux de compression à utiliser :

```
$ xtightvncviewer -compresslevel 3
```

La valeur 3 donne en pratique de bons résultats sans trop dégrader la qualité de l'image.

Les autres possibilités pour améliorer la fluidité sont :

- réduire la taille de « l'écran » de la GLAP-Box en 800x600 si ce n'est déjà le cas ( XFCE > Paramètres > Gestionnaire des Paramètres > Affichage ).
- Utiliser une connexion VNC non sécurisée par SSH (la sécurisation SSH ralentit la vitesse de transmission du réseau : entre rapidité et sécurité, il faut choisir)
- Utiliser une connexion wifi norme N à 300 Mb/s (contre 54 Mb/s pour la norme g classique).

## *Evaluer la bande passante du réseau lors de l'accès VNC*

Pour avoir une idée précise de ce qui se passe sur le réseau lors de l'accès VNC, il est utile d'avoir une visualisation de la bande passante. On peut faire cela aussi bien sur le poste fixe que sur la GLAP-Box elle-même.

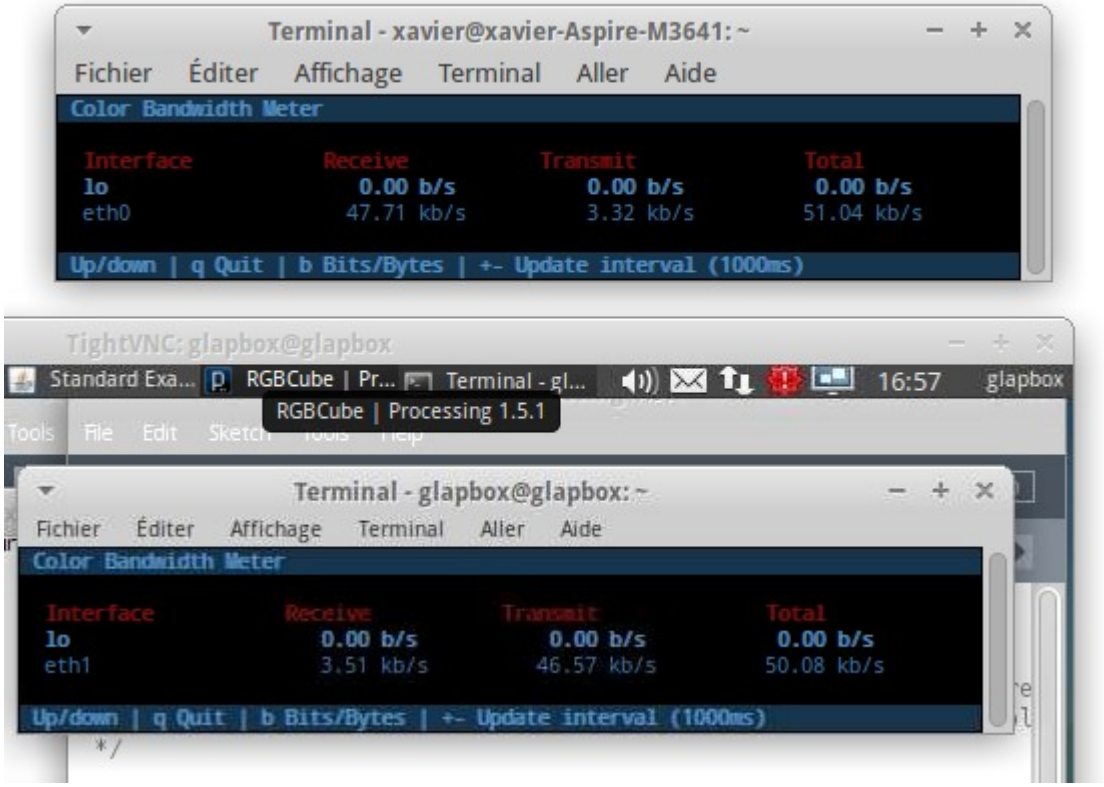
Sous Xubuntu, un utilitaire pratique qui fait cela est le programme en ligne de commande **cbm**. On commence par l'installer si ce n'est déjà fait soit sur le poste fixe, soit sur la GLAP-Box (déjà installé par défaut) :

```
$ sudo apt-get install cbm
```

Ensuite, on lance le programme avec la commande :

```
$ cbm
```

On obtient un affichage de la bande passante en bits/secondes ou Bytes/secondes (appuyer sur B pour basculer d'une unité vers l'autre) pour chaque interface réseau. Voici un exemple simultané de **cbm** sur le poste fixe et la GLAP-Box :

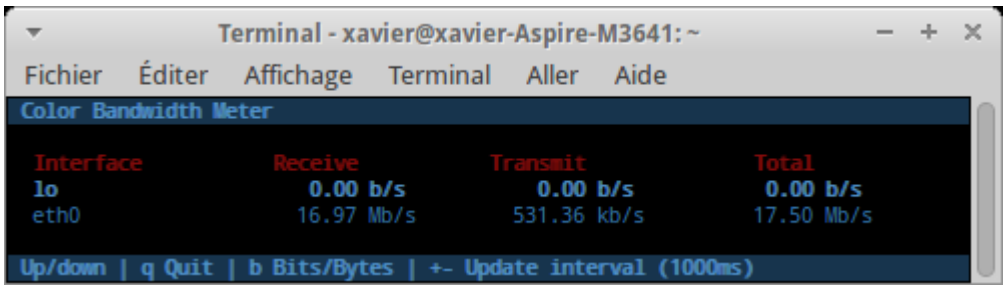

Terminal - xavier@xavier-Aspire-M3641: ~
Fichier Éditer Affichage Terminal Aller Aide
Color Bandwidth Meter
Interface Receive Transmit Total
lo 0.00 b/s 0.00 b/s 0.00 b/s
eth0 47.71 kb/s 3.32 kb/s 51.04 kb/s
Up/down | q Quit | b Bits/Bytes | +- Update interval (1000ms)
TightVNC: glapbox@glapbox
Standard Exa... RGBCube | Pr... Terminal - gl... 16:57 glapbox
RGBCube | Processing 1.5.1
Tools File Edit Sketch
Terminal - glapbox@glapbox: ~
Fichier Éditer Affichage Terminal Aller Aide
Color Bandwidth Meter
Interface Receive Transmit Total
lo 0.00 b/s 0.00 b/s 0.00 b/s
eth1 3.51 kb/s 46.57 kb/s 50.08 kb/s
Up/down | q Quit | b Bits/Bytes | +- Update interval (1000ms)
*/

On constate logiquement :

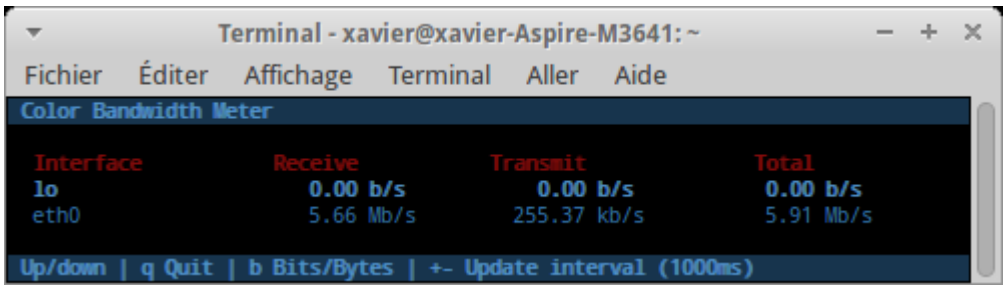
- que le débit global est sensiblement le même sur les 2 postes
- que le débit en émission d'un côté correspond au débit en réception de l'autre côté
- que VNC utilise une communication bi-directionnelle

Truc d'utilisation : réduire la fenêtre Terminal où s'exécute cbm et la mettre « toujours au premier plan » (clic droit sur la barre de fenêtre). Comme çà on visualise la bande passante en temps réel.

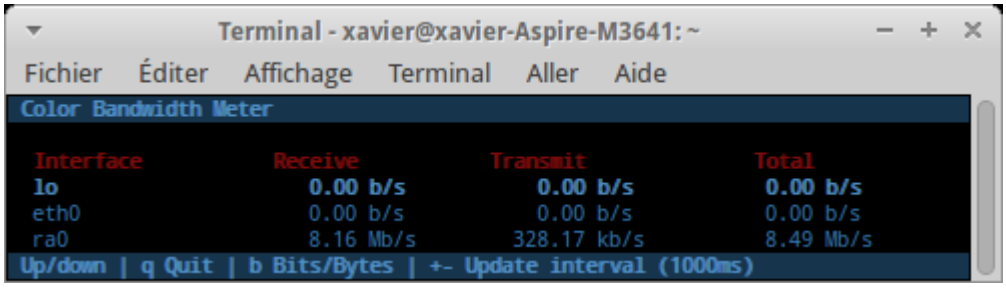
A titre indicatif, voici le chiffre de débit obtenu avec **xtightvncviewer** sans compression pour RGBCube dans Processing :


Terminal - xavier@xavier-Aspire-M3641: ~
Fichier Éditer Affichage Terminal Aller Aide
Color Bandwidth Meter
Interface Receive Transmit Total
lo 0.00 b/s 0.00 b/s 0.00 b/s
eth0 16.97 Mb/s 531.36 kb/s 17.50 Mb/s
Up/down | q Quit | b Bits/Bytes | +- Update interval (1000ms)

A titre indicatif, voici le chiffre de débit obtenu avec **xtightvncviewer** avec compression 3 pour RGBCube dans Processing (ce qui divise le débit par 2 !) :


Terminal - xavier@xavier-Aspire-M3641: ~
Fichier Éditer Affichage Terminal Aller Aide
Color Bandwidth Meter
Interface Receive Transmit Total
lo 0.00 b/s 0.00 b/s 0.00 b/s
eth0 5.66 Mb/s 255.37 kb/s 5.91 Mb/s
Up/down | q Quit | b Bits/Bytes | +- Update interval (1000ms)

Toujours sur un réseau en wifi N 300Mbps, voici le débit obtenu pour le RGB Cube de Processing avec compression 3 :


Terminal - xavier@xavier-Aspire-M3641: ~
Fichier Éditer Affichage Terminal Aller Aide
Color Bandwidth Meter
Interface Receive Transmit Total
lo 0.00 b/s 0.00 b/s 0.00 b/s
eth0 0.00 b/s 0.00 b/s 0.00 b/s
ra0 8.16 Mb/s 328.17 kb/s 8.49 Mb/s
Up/down | q Quit | b Bits/Bytes | +- Update interval (1000ms)

On obtient un gain de près de 50 % de la bande passante par rapport au réseau éthernet filaire.

Il y a d'autres programmes qui permettent cela : tout est question de goût.

### *Et aussi : sécuriser l'accès VNC par SSH*

Noter qu'il est également possible de se connecter à la GLAP-Box en VNC de façon sécurisée avec tunnellisation SSH : la procédure est décrite dans le document « GLAP-Box from Scratch ».

Cependant, cette sécurisation entraîne un ralentissement de la connexion et n'est utile qu'en cas de réseau ouvert. Sur un réseau local fermé dédié, et lorsqu'on recherche la vitesse de connexion en priorité, cette sécurisation n'est pas indispensable.

## Activation d'une auto-connexion de la GLAP-Box en wifi

Lors de la première connexion à la GLAP-Box, il est indispensable de se connecter en Ethernet (filaire), seule façon d'avoir un accès automatique au réseau local.

Une fois cette première connexion faite, on se connectera au réseau wifi manuellement depuis la

fenêtre VNC puis aller dans XFCE > Paramètres > Connexion Réseau > Onglet sans-fil > choisir la connexion voulue et :

- cocher Connecter automatiquement
- et cocher Accessible aux autres utilisateurs

Au prochain démarrage, la connexion wifi devrait être disponible automatiquement si le réseau est disponible. Si ce n'est pas le cas, reconnecter en filaire éthernet et vérifier ce qui ne va pas.

## Dépannage de la connexion réseau de la GLAP-Box

- Pour tout problème en wifi avec la GLAP-Box, avoir le réflexe de se connecter en réseau filaire Ethernet, solution de dépannage.
- Si on n'arrive pas également en Ethernet, on pourra essayer de simplement connecter un clavier à la GLAP-Box et faire un appui sur entrée : ceci relance le démarrage lorsqu'il se bloque suite à un arrêt brutal par exemple.
- Si malgré tout, on n'obtient aucune connexion automatique, il faut connecter un écran sur la sortie VGA + souris + clavier pour vérifier ce qui se passe... mais ceci ne doit quasiment jamais arriver à l'usage.

## Trucs pratiques d'utilisation de l'accès VNC à connaître

- Il est possible, par VNC, de copier/coller directement du texte depuis le poste fixe vers la GLAP-Box. Ainsi, par exemple, une ligne de commande dans un document sur le poste fixe pourra être copiée puis collée ensuite dans la fenêtre VNC. Ceci est très pratique. La même chose fonctionnera pour copier/coller un code Arduino, Processing, etc... (**à condition d'ouvrir le code depuis un éditeur texte simple, comme Gedit et pas depuis IDE Arduino ou Procesing !** ) offrant une souplesse de développement maximale. L'inverse ne semble pas fonctionner et il vaudra miexu se rabattre sur le partage de fichier NFS.

## *Mise en place du partage de fichier NFS entre la GLAP-Box et le poste fixe*

### *Présentation*

La GLAP-Box permet la mise en place d'un partage de fichier très simplement à partir d'un autre poste sous Xubuntu. Cette fonctionnalité est très pratique car elle permet de copier/coller des fichiers depuis le poste fixe vers la GLAP-Box et inversement très simplement, comme on le ferait dans un simple dossier du poste fixe lui-même.

Techniquement, la GLAP-Box est préconfigurée en serveur de fichier NFS et il est dès lors possible de la « monter » dans l'arborescence du système du poste fixe. La GLAP-Box devient alors accessible comme un simple dossier du système.

Cette fonction est très pratique car elle permet notamment de transférer des codes vers la GLAP-Box, des librairies, mais également des fichiers sonores. Cette fonction permet également de copier sur le poste fixe des fichiers présent sur la GLAP-Box.

Cette fonction fonctionne aussi bien en connexion ethernet filaire que par wifi : ainsi, on pourra transférer ou récupérer des dossiers ou fichiers sur un robot mobile embarquant une GLAP-Box, sans fil et à distance en toute simplicité !

On reprend ici la procédure présentée dans le document « GLAP-Box from Scratch » :

## *Accéder au partage de fichier depuis le client (= le poste fixe)*

### Installation des paquets utiles

Installer dans synaptic sur le poste fixe:

- nfs-common

### Création d'un point de montage sur le client

Dans une console, créer un répertoire pour le montage nfs avec la commande :

```
$ sudo mkdir /mnt/nfs
```

### Montage du serveur (embarqué) sur le client (fixe)

On utilise ici l'adresse du serveur : 192.168.1.x C'est l'adresse obtenue par nmap comme vu précédemment ou encore par ifconfig sur la Glap-Box.

```
$ sudo mount -t nfs 192.168.1.x:/home/glapbox/ /mnt/nfs
```

ou

```
$ sudo mount -t nfs -o rw 192.168.1.x:/home/glapbox/ /mnt/nfs
```

Voilà, si pas de message d'erreur, c'est bon... Ne pas oublier le / avant le nom du chemin de montage après l'adresse...

A noter, pour automatiser le montage NFS depuis le poste client, on pourra utiliser un script comme par exemple :

```
#!/bin/bash
# script montage NFS
# par XH - Aout 2011

echo "--- script de Montage NFS côté client ---" $disque

read -p "Quelle est l'adresse du serveur NFS (192.168.x.x ou équiv.) ? " adresse #saisie de l'adresse
echo "L'adresse IP du serveur NFS à monter est :" $adresse

read -p "Quel est le répertoire du /home du serveur NFS à monter sur le client (/xavier ou /glapbox/Bureau ou équiv) ?" repertoire #saisie du répertoire
echo "Le répertoire du /home du serveur NFS à monter sur le client est :" $repertoire

echo "Le script va monter "$adresse":/home"$repertoire"/ sur /mnt/nfs"

read -p "<OK>" # pour attendre entrée pour continuer
sudo mount -t nfs -o rw $adresse:/home$repertoire/ /mnt/nfs
```

```
echo "Opération terminée"

read -p "<OK>" # pour attendre entrée pour sortir

exit 0 ;
```

#### Test

Saisir dans la console pour voir si le montage nfs est bien en place :

```
$ mount | grep nfs
```

ce qui donne

```
nfsd on /proc/fs/nfsd type nfsd (rw)
192.168.1.03:/home/xavier/ on /mnt/nfs type nfs (rw,addr=192.168.1.03)
```

#### Utilisation

Il suffit à présent d'aller dans répertoire /mnt/nfs : on doit voir le contenu de la GLAP-Box ! On peut dès lors réaliser des échanges de fichiers entre la GLAP-Box et le PC de contrôle en toute simplicité !

On pourra également créer un lanceur sur le bureau du client vers le répertoire /mnt/nfs pour accéder facilement au contenu du répertoire. A noter que pour ouvrir un dossier à partir d'un lanceur, il faudra faire :

- sous Ubuntu : nautilus /rep
- sous xubuntu : thunar /rep soit dans notre cas, thunar /mnt/nfs

## *Test des principales fonctionnalités utiles de la GLAP-Box*

Ici sont rassemblées toute une série de procédures de test des principales fonctionnalités de la GLAP-Box, ce qui permet

## Tests des fonctions du système embarqué

### *Test de la lecture de fichiers sons*

### *Test de la synthèse vocale*

#### Test de la synthèse vocale avec espeaker / Gspeaker

#### Test de la synthèse vocale avec pico TTS

### *Tests vidéos*

**Test de la capture webcam standard**

**Test de la capture webcam « haute vitesse »**

**Test du retour vidéo**

### *Tests audio*

**Test de la capture audio**

**Test du retour audio**

## Test du logiciel Arduino et de la communication série

### *Test de la programmation de la carte Arduino*

### *Test de la communication série avec la carte Arduino*

> prog miroir...

### *Test de la conversion analogique numérique*

## Tests du logiciel Processing et des librairies installées

Afin de pouvoir facilement gérer les fonctionnalités du système à partir de Processing, je vous propose la librairie Glapbox pour Processing qui intègre toute une série de fonctions faciles à utiliser pour accéder directement aux programmes du système tels que la synthèse vocale, la lecture de fichier sons, etc..

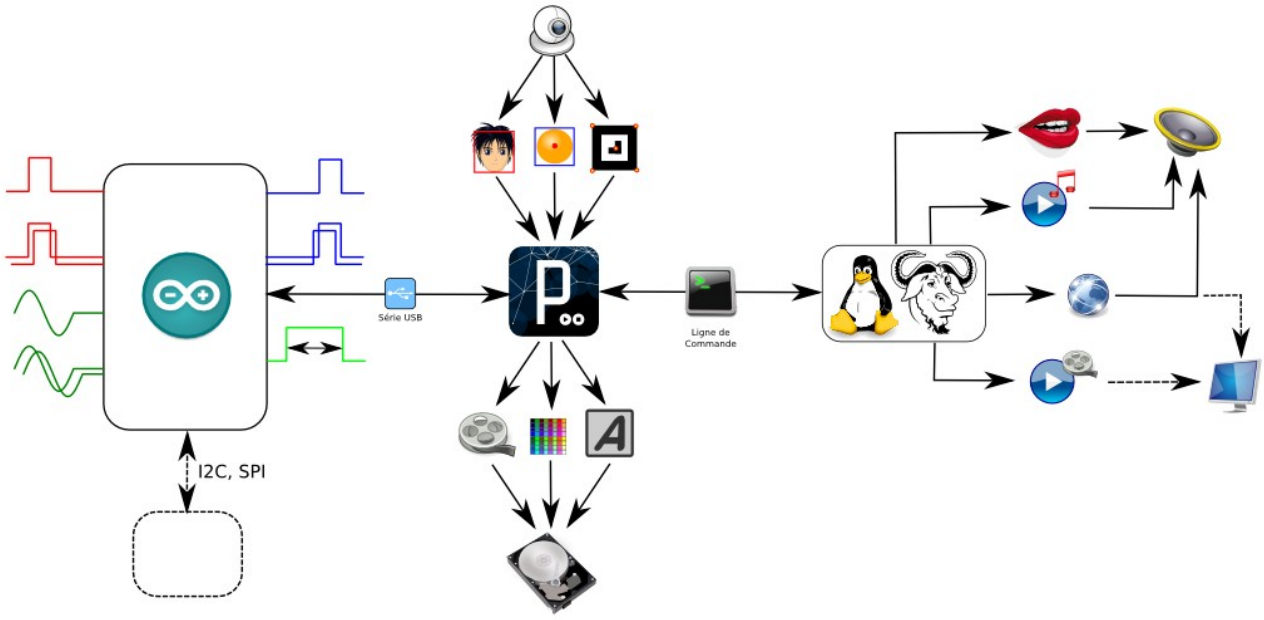
**Remarque**

Tous les codes fournis dans cette documentation fonctionnent directement sur la GLAP-Box par simple copier/coller et lancement de l'exécution côté Arduino ou Processing selon les cas.

Tous ces codes sont à utiliser « côté GLAP-Box » simplement par copier/coller du poste fixe vers la GLAP-Box ou sur la GLAP-Box elle-même via l'accès VNC depuis le poste fixe.

### *Intro : synthèse des possibilités fonctionnelles de la GLAP-Box*

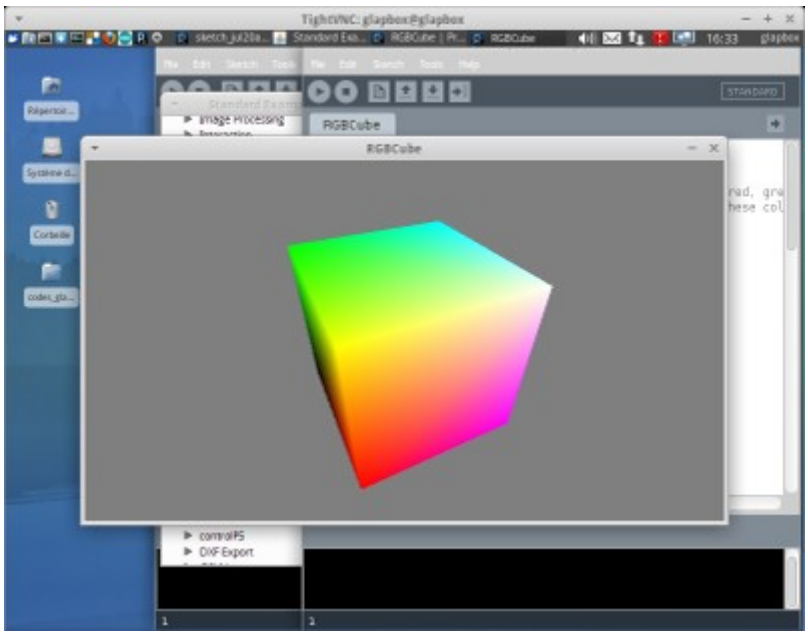
Voici sur ce schéma la synthèse des possibilités fonctionnelles de la GLAP-Box :




### *Test simple de Processing*

Ce test de base de Processing est très facile à faire et permet de s'assurer que Processing fonctionne normalement :

- se connecter à la GLAP-Box (voir ici : Accès au bureau distant par VNC )
- clic sur l'icône Processing dans le tableau supérieur du bureau de la GLAP-Box
- une fois Processing activé, aller dans Examples > 3D > Form > RGBCube (ou tout autre code de son choix...)
- puis clic sur le bouton d'exécution : on doit obtenir la fenêtre graphique de Processing suivante :




## *Test du contrôle du système par ligne de commande programmée*

### Test du déclenchement de la synthèse vocale par programme

*Matériel*

Ce test nécessite une paire d'enceintes ou un casque connecté à la GLAP-Box.

*Explications*

La librairie Glapbox pour Processing permet d'accéder à 2 moteurs de synthèse vocale présents sur la GLAP-Box : espeak et picoTTS. Les fonctions disponibles avec la librairie Glapbox sont :

- void direTexte(java.lang.String texte, java.lang.String moteurTTS, boolean flagWait)
- void direTexteEspeak(java.lang.String texte)
- void direTexteEspeakNoWait(java.lang.String texte)
- void direTextePico(java.lang.String texte)
- void direTextePicoNoWait(java.lang.String texte)

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
//-- Programme Processing pour la GLAP-Box
// par X. HINAULT - Tous droits réservés - GPL v3 - 2012

// Test de la synthèse vocale

import monclubelec.glapbox.*; // importe la librairie Glapbox qui comporte tout plein de fonctions utiles...

Glapbox glapbox; // Déclare objet Glapbox qui donne accès aux fonctions de la librairie

void setup () {

  // initialisation objet Glapbox
  glapbox=new Glapbox(this); // initialise objet Glapbox
```

```
}

void draw () {

  String texte="Bonjour à tous ! Je mappelle pico et je sais parler !";

  glapbox.direTexte(texte, "PICOTTS", true); // synthèse vocale avec PICOTTS - attend fin synthese

  texte="Bonjour à tous ! Je mappelle ispik et je sais aussi parler !";
  glapbox.direTexte(texte, "ESPEAK", true); // synthèse vocale avec PICOTTS - attend fin synthese

  texte="Ouais. Pas mal. Mais moi, pico, je parle beaucoup mieux que toi, mon cher ispik";

  glapbox.direTexte(texte, "PICOTTS", true); // synthèse vocale avec PICOTTS - attend fin synthese

  texte="Sa reste  a voir, ma chaire pico.";
  glapbox.direTexte(texte, "ESPEAK", true); // synthèse vocale avec PICOTTS - attend fin synthese

  while(true); // stoppe draw

}
```

Truc : éviter les ' dans les chaînes et modifier parfois l'orthographe pour avoir la prononciation voulue. Ne pas hésiter également, pour ralentir le débit,  à bien ponctuer la chaîne avec des , ou des .

## *Test du lancement de la lecture d'un fichier son depuis Processing*

*Matériel*

Ce test nécessite une paire d'enceintes ou un casque connecté à la GLAP-Box.

*Explications*

Il est possible de lire n'importe quel fichier son sur la GLAP-Box à l'aide de la librairie Processing Glapbox qui permet d'utiliser notamment aplay, mplayer. On pourra donc sonoriser ses applications embarquées très facilement avec des fichiers téléchargés sur internet notamment.

Un répertoire contenant quelques exemples de sons est présent sur le bureau de la GLAP-Box

Les fonctions disponibles avec la librairie Glapbox sont :

- void joueSon(java.lang.String cheminSonIn, java.lang.String fichierSonIn, java.lang.String lecteurSon, boolean flagWait)
- void joueSonAplay(java.lang.String cheminSonIn, java.lang.String fichierSonIn)
- void joueSonAplayNoWait(java.lang.String cheminSon, java.lang.String fichierSon)
- void joueSonMplayer(java.lang.String cheminSonIn, java.lang.String fichierSonIn)
- void joueSonMplayerNoWait(java.lang.String cheminSon, java.lang.String fichierSon)

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
//-- Programme Processing pour la GLAP-Box
// par X. HINAULT - Tous droits réservés - GPL v3 - 2012
```

```
// Test lecture fichier son

import monclubelec.glapbox.*; // importe librairie Glapbox qui comporte tout plein de fonctions utiles

Glapbox glapbox; // Déclare objet Glapbox qui donne accès aux fonctions de la librairie

String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/

void setup () {

  // initialisation objet Glapbox
  glapbox=new Glapbox(this); // initialise objet Glapbox


}

void draw () {

  glapbox.joueSon(homePath+"Bureau/mes_sons/", "r2d2.mp3", "MPLAYER", true); // lit le fichier son

  glapbox.joueSon(homePath+"Bureau/mes_sons/", "lightsaber_turn_on.wav", "APLAY", false); // lit le fichier son
  //--- attention : aplay ne lit pas fichier mp3...

  while(true); // stoppe draw

}
```

Pour info :

- mplayer est très polyvalent, mais peut créer un délai de fin de lecture.
- aplay est plus rapide en déclenchement/fermeture, donc intéressant pour lectures courtes

Remarque : si les fichiers n'existent pas, aucun son n'est lu.

## *Test de l'ouverture d'une webradio depuis Processing (ou d'une page web)*

### *Matériel*

Ce test nécessite une paire d'enceintes ou un casque connecté à la GLAP-Box.

On suppose également que la GLAP-Box est connectée à internet via le routeur du réseau.

### *Explications*

Les fonctions disponibles avec la librairie Glapbox sont :

- void lanceFirefox(java.lang.String adresseSite)
- void stopFirefox()

### *Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
//-- Programme Processing pour la GLAP-Box
// par X. HINAULT - Tous droits réservés - GPL v3 - 2012
```

```
// Test lancement URL site avec Firefox (webradio)

import monclubelec.glapbox.*; // importe librairie Glapbox qui comporte tout plein de fonctions
utiles

Glapbox glapbox; // Déclare objet Glapbox qui donne accès aux fonctions de la librairie

//String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/

void setup () {

  // initialisation objet Glapbox
  glapbox=new Glapbox(this); // initialise objet Glapbox

  glapbox.lanceFirefox ("http://mfmradio.fr/radio/player/mfm-radio-en-vacances");

}

void draw () {


}
```

On pourra utiliser cette fonction pour sélectionner une webradio à la demande à partir de boutons poussoir par exemple, transformant la Glap-Box en « web radio ».

#### A faire..

Test de la récupération de l'adresse IP

une fonction de la forme getIP(« eth »)... Revoir fonction déjà faite...

Test de la récupération de l'info utile à partir de la réponse d'un programme lancé en ligne de commande : exemple d'un programme de calcul de l'élévation solaire.

## *Test d'enregistrement dans des fichiers*

#### Test de l'enregistrement de données dans un fichier texte

*Matériel*

Aucun matériel nécessaire en dehors de la GLAP-Box.

*Explications*

On teste ici l'enregistrement de données dans un fichier texte. A intervalle régulier, on écrit des valeurs dans un fichier texte au format dit « CSV » qui sera ensuite facilement utilisable dans un serveur.

Cette fonctionnalité sera très pratique pour réaliser des enregistrements de données à partir de mesures faites par une carte Arduino par exemple.

On utilise ici la fonction suivante de la librairie Glapbox pour Processing :

- java.io.PrintWriter ouvreFichierTexte(java.lang.String cheminAbsoluFichierIn)

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
//-- Programme Processing pour la GLAP-Box
// par X. HINAULT - Tous droits réservés - GPL v3 - 07/2011 - MAJ 07/2012

// Test de l'écriture de données dans un fichier texte

import monclubelec.glapbox.*; // importe librairie Glapbox qui comporte tout plein de fonctions utiles

//--- Objets utiles ---

Glapbox glapbox; // Déclare objet Glapbox qui donne accès aux fonctions de la librairie

PrintWriter output = null; // initialise un objet PrintWriter pour stocker le flux de donnée à écrire dans le fichier

String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/
String cheminAbsoluFichier=homePath+"Bureau/trans/positions.txt";
// le fichier doit exister - chemin entier obligatoire

long seconde0=0, comptSeconde=0; // variables des secondes

boolean debug=true; // activation du debug

void setup() {

  // initialisation objet Glapbox
  glapbox=new Glapbox(this); // initialise objet Glapbox

  // Crée un nouveau fichier - efface fichier si existe déjà !
  cheminAbsoluFichier=homePath+"Bureau/trans/positions"+"_"+year()+"_"+month()+"_"+day()+"_"+hour()+"_"+minute()+"_"+second()+".txt";
  output = createWriter(cheminAbsoluFichier);
  // cette ligne crée un fichier vide au nom unique horodaté
  // il est parfois plus utile de créer manuellement un fichier...dans lequel les différents enregistrements s'enchaîneront

   output=glapbox.ouvreFichierTexte(cheminAbsoluFichier);   // ouvre le fichier existant dans un PrintWriter- le fichier doit exister - chemin entier obligatoire
   output.println("------------ Debut Nouvel enregistrement -------------"); // Ajoute la ligne au fichier
   output.println("débuté le " + day()+"/"+month()+"/"+year()+" à "+hour()+":"+minute()+":"+second());
   output.println("------------ Format CSV  -------------"); // Ajoute la ligne au fichier
   output.println("secondes; valeur1; valeur2;"); // Ajoute la ligne au fichier

   output.flush(); // Ecrit les données dans le fichier

} // fin setup()

void draw() {

  if (second()!=seconde0) { // si une seconde écoulée

    seconde0=second(); // retient dernière seconde prise en compte

    comptSeconde=comptSeconde+1; //incrémente la variable de comptage seconde

    point(mouseX, mouseY); // trace un point dans la fenetre Graphique

    output.print(comptSeconde+";"); //Ecrit la seconde courante dans le PrintWriter
    output.print(mouseX+";"); //Ecrit les coordonnées courantes de la souris dans le PrintWriter
    output.print(mouseY+";");
    output.println(); // Ajoute saut de ligne

    //--- affichage dans la console ---
    if (debug) {
      print("Ajoute au fichier la ligne : ");
      print(comptSeconde+";"); //Ecrit la seconde courante dans le PrintWriter
      print(mouseX+";"); //Ecrit les coordonnées courantes de la souris dans le PrintWriter
      print(mouseY+";");
      println(); // Ajoute saut de ligne

    } // fin if debug

    output.flush(); // Ecrit les données dans le fichier
```

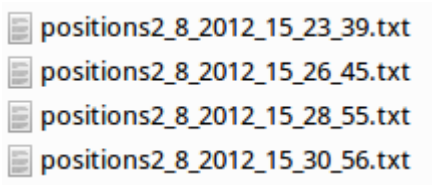
```
  } // fin if

} // fin draw()

void keyPressed() { // si une touche est appuyée on sort du programme

  output.close(); // Termine l'enregistrement en fermant le fichier
  exit(); // Stoppe le programme
}
```

*Fonctionnement, principe d'utilisation*

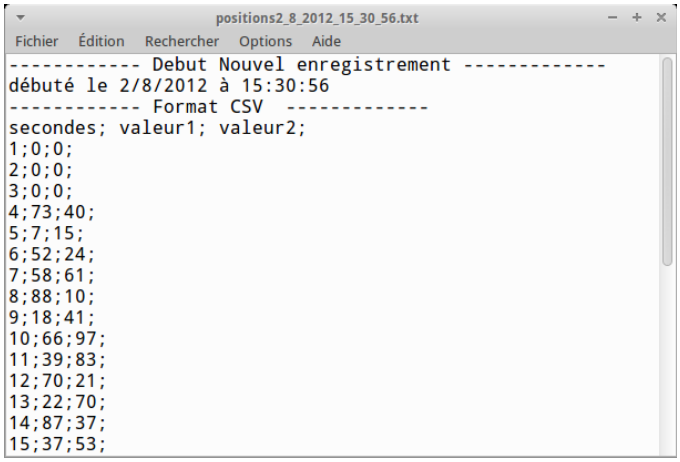
Une fois le programme lancé, tant que l'on déplace la souris dans la fenêtre Processing, la position est enregistrée chaque seconde. On peut envisager d'enregistrer de la sorte tout ce que l'on veut au format texte, des chaînes en provenance d'Arduino sur le port série notamment. On peut ainsi réaliser du datalogging simplement, ou créer des fichiers de suivis d'évènements.

Noter la possibilité d'horodater facilement le nom du fichier.

On obtient ensuite un fichier dans le répertoire utilisé qui est accessible par VNC ou NFS :



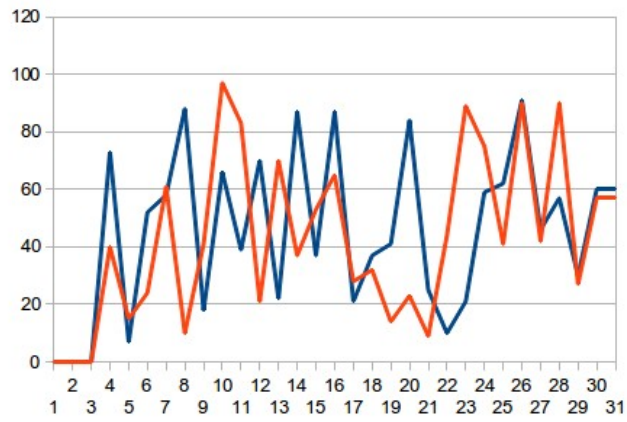

On ouvre ce fichier en double cliquant sur le nom et on obtient :




Ensuite il suffit de faire un copier/coller dans un tableur (LibreOffice bien sûr) : lors du « coller », il est possible de fixer le séparateur utilisé, ici le point virgule. On obtient alors :

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | |
| 2 | 2 | 0 | 0 | |
| 3 | 3 | 0 | 0 | |
| 4 | 4 | 73 | 40 | |
| 5 | 5 | 7 | 15 | |
| 6 | 6 | 52 | 24 | |
| 7 | 7 | 58 | 61 | |
| 8 | 8 | 88 | 10 | |
| 9 | 9 | 18 | 41 | |
| 10 | 10 | 66 | 97 | |
| 11 | 11 | 39 | 83 | |
| 12 | 12 | 70 | 21 | |
| 13 | 13 | 22 | 70 | |
| 14 | 14 | 87 | 37 | |
| 15 | 15 | 37 | 53 | |
| 16 | 16 | 87 | 65 | |
| 17 | 17 | 21 | 28 | |
| 18 | 18 | 37 | 32 | |
| 19 | 19 | 41 | 14 | |
| 20 | 20 | 84 | 23 | |

Il est alors très facile et simple d'obtenir le graphique correspondant aux données enregistrées, ce qui donne :




On pourra donc très simplement utiliser les données enregistrées, les journaliser dans des fichiers différents, inclure les graphiques dans des documents, etc... Le tout uniquement avec des ressources libres et opensource !

Un potentiel de possibilités donc assez important, d'autant que les données pourront être des mesures en provenance d'Arduino. On pourra donc de la sorte très facilement par exemple enregistrer les sorties de capteurs et les analyser secondairement « à froid ».

Truc d'utilisation : il est possible de consulter le fichier de données alors que l'enregistrement est actif. L'enregistrement continuera et il suffira de mettre à jour le fichier (très facile à faire si on ouvre le fichier texte avec un navigateur web tel que Firefox) pour visualiser les dernières données enregistrées.

A faire :

test de la vitesse max d'enregistrement (= en théorie, celle du framerate du programme)

#### Test de l'enregistrement de fichier image à la demande

*Matériel*

Nécessite une webcam connectée à la GLAP-Box.

*Explications*

On teste ici l'enregistrement d'images à la demande, ici à intervalle régulier. A intervalle régulier, on enregistre dans un fichier l'image capturée en provenance du flux vidéo. En pratique, une telle capture pourra servir à enregistrer des mouvements ultra-lents pour les accélérer ensuite. On pourra utiliser ce principe également pour réaliser des captures d'une image lors de la survenue d'un événement déclencheur, notamment à partir de la carte Arduino.

On utilise ici la librairie Gsvideo pour lire une frame en provenance de la webcam et on enregistre cette image dans un fichier.

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
// Programme processing
// généré avec le générateur de code Processing
// du site www.mon-club-elec.fr
// par X. HINAULT - tous droits réservés

// Programme écrit le : 14/9/2011 - MAJ 07/2012

// ------- Licence du code de ce programme : GPL v3-----


/////////////// Description du programme ////////////
// Utilise un/des objets PImage (conteneur d'image .jpeg .gif .png et .tga)
// Utilise le clavier
// Utilise la librairie GSVideo de capture et lecture vidéo

/*
> Le programme capture et affiche le flux vidéo en provenance d'une webcam
> L'appui sur la touche e lance l'enregistrement des images à intervalle régulier fixé par le progr
amme.
> L'appui sur la touche s stoppe l'enregistrement des images
> L'appui sur la touche l lance la lecture des images enregistrées
*/

// XXXXXXXXXXXXXXXXXXXXXX ENTETE DECLARATIVE XXXXXXXXXXXXXXXXXXXXXX

// inclusion des librairies utilisées

import codeanticode.gsvideo.*; // importe la librairie vidéo GSVideo qui implémente GStreamer pour
Processing (compatible Linux)
// librairie comparable à la librairie native vidéo de Processing (qui implémente QuickTime..)- Voi
r Reference librairie Video Processing
// cette librairie doit être présente dans le répertoire modes/java/libraries du répertoire Process
ing (1-5)
// voir ici : http://gsvideo.sourceforge.net/
// et ici : http://codeanticode.wordpress.com/2011/05/16/gsvideo-09-release

// déclaration objets

PImage img1, img2; // déclare un/des objets PImage (conteneur d'image)

GSCapture cam1; // déclare un objet GSCapture représentant une webcam
// L'objet GSCapture étend PImage - se comporte comme un conteneur des frames issues de la webcam

String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/
String cheminRepertoireEnregistrement=homePath+"/Bureau/trans/captureImages"+"_"+year()+"_"+month()
+"_"+day()+"_"+hour()+"_"+minute()+"_"+second()+"/"; // chemin absolu du répertoire
d'enregistrement
//--- nom horodaté pour nom unique - ne pas oublier le / de fin de chemin

// déclaration variables globales

// variable pour la taille de la capture video
int widthCapture=320; // largeur capture
int heightCapture=240; // hauteur capture
int fpsCapture=20; // framerate (image/secondes) pour la capture video
```

```
long delai=1000; // délai en millisecondes entre 2 enregistrement d'image
long millis0=0; // variable pour mémoriser dernier valeur millis() pris en compte

long comptImg=0; // variable comptage image
long comptImgPlay=0; // variable comptage image lecture
String numeroImg="00000"; // chaine pour numéro image au format 00001

int vitessePlay=100; // durée pause entre 2 lecture images enregistrées

boolean flagRecord=false; // drapeau activation enregistrement
boolean flagPlay=false; // drapeau activation lecture

// XXXXXXXXXXXXXXXXXXXXX  Fonction SETUP XXXXXXXXXXXXXXXXXXXXX

void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

        // ---- initialisation paramètres graphiques utilisés
        colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
        fill(0,0,255); // couleur remplissage RGB - noFill() si pas de remplissage
        stroke (0,0,0); // couleur pourtour RGB - noStroke() si pas de pourtour
        rectMode(CORNER); // origine rectangle : CORNER = coin sup gauche | CENTER : centre
        imageMode(CORNER); // origine image : CORNER = coin sup gauche | CENTER : centre
        ellipseMode(CENTER); // origine cercles / ellipses : CENTER : centre (autres : RADIUS,
CORNERS, CORNER
        //strokeWeight(0); // largeur pourtour
        frameRate(15);// Images par seconde - The default rate is 60 frames per second

        // --- initialisation fenêtre de base ---
        size(widthCapture*2, heightCapture); // ouvre une fenêtre xpixels  x ypixels
        background(0,0,0); // couleur fond fenetre

// --- initialisation des objets et fonctionnalités utilisées ---


        //======== Initialisation Objets GSVideo (capture et/ou lecture video =========

        // GSCapture(this, int requestWidth, int requestHeight, [int frameRate], [String
sourceName], [String cameraName])
        cam1 = new GSCapture(this, widthCapture, heightCapture,"v4l2src","/dev/video0",fpsCapture);
// Initialise objet GSCapture désignant webcam
        // largeur et hauteur doivent être compatible avec la webcam - typiquement 160x120 ou
320x240 ou 640x480...
        // Meilleurs résultats avec framerate webcam entre 20 et 30 et frameRate programme idem ou
multiple plus grand (40 pour 20 par ex)
        // la liste des webcam installées sous Ubuntu (Gnu/Linux) est donnée par la commande :
ls /dev/video*

        // cam1.play();  // démarre objet GSCapture = la webcam - version GSVideo avant 0.9
        cam1.start();  // démarre objet GSCapture = la webcam - version GSVideo après 0.9


} // fin fonction Setup

// XXXXXXXXXXXXXXXXXXXXX Fonction Draw XXXXXXXXXXXXXXXXXXXX

void  draw() { // fonction exécutée en boucle

// Code type capture GSVideo - utilisation possible aussi de captureEvent()

  if (cam1.available() == true) { // si une nouvelle frame est disponible sur la webcam
    cam1.read(); // acquisition d'un frame
    img1=cam1.get(); // récupère l'image GS video dans Pimage

    image(img1, 0, 0); // affiche image

     // --- création chaine numéro image au format 00000 ---
     if (comptImg>=10000) numeroImg=""+comptImg;
     if (comptImg<10000) numeroImg="0"+comptImg;
     if (comptImg<1000) numeroImg="00"+comptImg;
     if (comptImg<100) numeroImg="000"+comptImg;
     if (comptImg<10) numeroImg="0000"+comptImg;


    if ((millis()>millis0+delai) && (flagRecord==true)) { // si le délai voulu est écoulé et
enregistrement activé

      millis0=millis(); // mémorise millis() courant
```

```
      print("Millis()="+millis0+ " | ");

      image(img1,widthCapture,0); // affiche l'image capturée séparément du flux vidéo

      img1.save(cheminRepertoireEnregistrement+"monimage-test-"+numeroImg+".jpg"); // enregistre le
fichier image
      //--- le répertoire est créé si il n'existe pas !
      println("Image "+ cheminRepertoireEnregistrement+"monimage-test-"+numeroImg+".jpg enregistrée
!");

      comptImg=comptImg+1; // incrémente variable comptage image


      // --- alternative possible ---
     // saveFrame(cheminRepertoireEnregistrement+"monimage-#####.jpg"); // enregistre le frame
courant de la fenêtre d'affichage actuelle
     // au format monimage-00001.jpg, monimage-00002.jpg, etc...
     // le numéro de frame est celui du frame courant depuis début programme et pas forcément
numéro progressif

    }

  } // fin if available

  if (flagPlay==true) { // si la lecture est activée

            //---- défile les images capturées ---


             // --- création chaine numéro image au format 00000 ---
             if (comptImgPlay>=10000) numeroImg=""+comptImgPlay;
             if (comptImgPlay<10000) numeroImg="0"+comptImgPlay;
             if (comptImgPlay<1000) numeroImg="00"+comptImgPlay;
             if (comptImgPlay<100) numeroImg="000"+comptImgPlay;
             if (comptImgPlay<10) numeroImg="0000"+comptImgPlay;

               //---- lecture du fichier à partir nom ---
               img2=loadImage(cheminRepertoireEnregistrement+"monimage-test-"+numeroImg+".jpg");

              image(img2,widthCapture,0); // affiche l'image
              delay(vitessePlay);

             //--- gestion comptImgPlay
           comptImgPlay++;  // incrémente variable

            if (comptImgPlay>=comptImg) { // quand toutes les images visualisées
                println(comptImgPlay+ " images visualisées : lecture terminée !");
                comptImgPlay=0;
                flagPlay=false;
              }

  } // fin flagPlay true


        // while(true); // stoppe boucle draw

} // fin de la fonction draw()

// XXXXXXXXXXXXXXXXXXXXX Autres Fonctions XXXXXXXXXXXXXXXXXXXXX

//------------ gestion évènement clavier ---------

void keyPressed() { // si une touche est appuyée

        if(key=='e') { // si touche e enfoncée

            flagRecord=true;
            println("Enregistrement activé !");

        }


        if(key=='s') { // si touche s enfoncée

            flagRecord=false;
            println("Enregistrement stoppé !");
```

```
        }

        if(key=='l') { // si touche l enfoncée = lance lecture

            flagRecord=false;
            println("Enregistrement stoppé !");

            flagPlay=true;
            println("Lecture enregistrement lancée !");


        }

} //--- fin si touche enfoncee


//------------- Fonction d'arret de Processing ----

public void stop(){ // fonction d'arrêt de Processing

        cam1.delete(); // efface l'objet GScapture

        super.stop(); // obligatoire

} // fin fonction stop()


//XXXXXXXXXXXXXXXXXX Fin du programme XXXXXXXXXXXXXXXXX
```

*Fonctionnement, principe d'utilisation*

Avant le lancement, on pourra fixer dans le code le délai de capture en millisecondes entre 2 images.

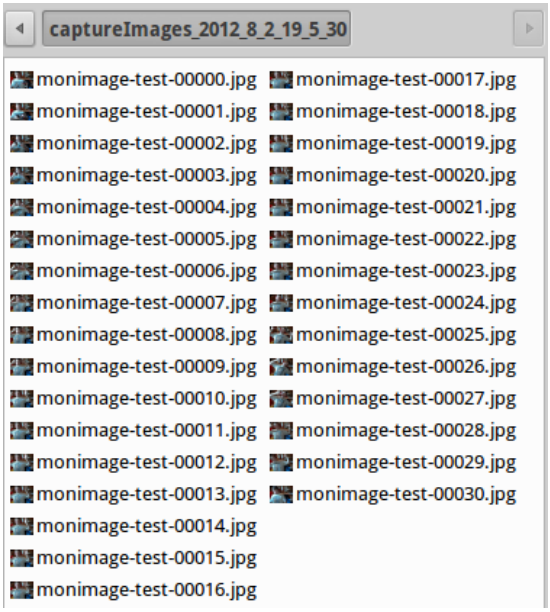
Au lancement, on obtient une fenêtre Processing divisée en 2 :

- à gauche, on visualise le flux vidéo direct
- à droite, on obtient visualise la capture/lecture des images

Le programme utilise :

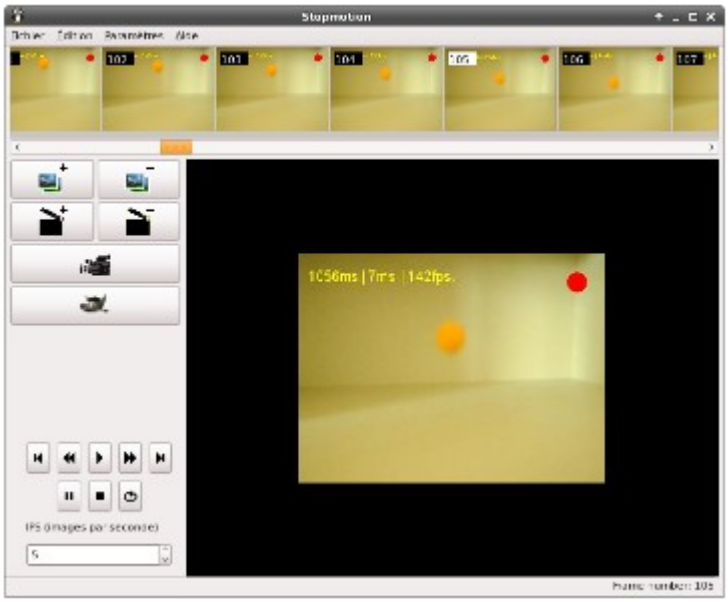
- l'appui sur la touche e pour lancer l'enregistrement
- l'appui sur la touche s pour stopper l'enregistrement. Noter que si on relance l'enregistrement, les images sont enregistrées à la suite.
- l'appui sur la touche p pour lire les images capturées

Une fois la capture terminée, on obtient un répertoire horodaté qui contient les fichiers images numérotés :




Pour manipuler facilement ces images, il existe plusieurs logiciel. Sous Xubuntu et donc sur la GLAP-Box, il existe stopmotion, accessible depuis le menu XFCE > Multimédia > Stopmotion.

Avec ce logiciel, de cliquer sur le bouton « ajouter des images », puis d'ouvrir le répertoire des images, de toutes les sélectionner et de lancer la lecture en fixant la vitesse de défilement. Très pratique. Ce logiciel permet également d'exporter les images dans un fichier vidéo.




Sinon, pour des manipulations plus avancées, on pourra aussi utiliser kino sous Ubuntu (sur poste fixe).

Cette technique de capture d'images sera particulièrement utile pour des captures d'images déclenchées par un événement ou encore des visualisations d'évènements lents.

### Test de l'enregistrement de capture vidéo standard

*Matériel*

Nécessite une webcam connectée à la GLAP-Box.

*Explications*

De la même façon qu'il est possible de déclencher depuis Processing l'acquisition d'images à la demande, il est également possible de réaliser des enregistrements vidéos à la demande, notamment sur évènements déclencheurs en provenance d'Arduino.

On utilise à nouveau ici la librairie GSVideo et l'objet GSMovieMaker qui permet de créer une vidéo en ajoutant chaque nouveau frame au fichier. L'encodage utilisé est paramétrable.

Cette fonction peut être intéressante sur des applications de « piège vidéo » où la présence d'une personne ou d'un animal déclenche la capture vidéo. Noter que chaque vidéo sera enregistrée dans un fichier horodaté.

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
// Programme processing
// généré avec le générateur de code Processing
// du site www.mon-club-elec.fr
// par X. HINAULT - tous droits réservés

// Programme écrit le : 15/9/2011. MAJ 07/2012

// ------- Licence du code de ce programme : GPL v3-----

/////////////// Description du programme ////////////
// Utilise le clavier
// Utilise la librairie GSVideo de capture et lecture vidéo

/*
> ce programme affiche un flux vidéo capturé par la librairie GSVideo
> l'appui sur la touche 'e' enregistre le flux vidéo dans un fichier
(Attention : efface l'enregistrement précédent)
> l'appui sur la touche 's' stoppe l'enregistrement

> la lecture du fichier vidéo se fera ensuite par double clic sur le nom du fichier

*/

// XXXXXXXXXXXXXXXXXXXXX ENTETE DECLARATIVE XXXXXXXXXXXXXXXXXXXXX

// inclusion des librairies utilisées

import codeanticode.gsvideo.*; // importe la librairie vidéo GSVideo qui implémente GStreamer pour
Processing (compatible Linux)
// librairie comparable à la librairie native vidéo de Processing (qui implémente QuickTime..)- Voi
r Reference librairie Video Processing
// cette librairie doit être présente dans le répertoire modes/java/libraries du répertoire Process
ing (1-5)
// voir ici : http://gsvideo.sourceforge.net/
// et ici : http://codeanticode.wordpress.com/2011/05/16/gsvideo-09-release

// déclaration objets

GSCapture cam1; // déclare un objet GSCapture représentant une webcam
// L'objet GSCapture étend PImage - se comporte comme un conteneur des frames issues de la webcam

GSMovieMaker maVideo; // crée un objet moviemaker pour créer la vidéo

// déclaration variables globales

PImage imgDest;

String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/
String cheminFichier=homePath+"/Bureau/trans/"; // chemin absolu du fichier
String nomFichier="video"+"_"+year()+"_"+month()+"_"+day()+"_"+hour()+"_"+minute()+"_"+second()
+".avi"; // nom de la vidéo horodaté = nom unique

//------ déclaration des variables de couleur utiles ----
```

```
int rouge=color(255,0,0);



// variable pour la taille de la capture video
int widthCapture=640; // largeur capture
int heightCapture=480; // hauteur capture
// plus la taille est grande et plus le fichier vidéo sera gros

int fpsCapture=20; // framerate (image/secondes) pour la capture video
//--- vérifier au préalablable dans Guvcview que le framerate courant de la webcam est réglé sur la même valeur

// --- plus le framerate est élevé et plus le fichier vidéo sera gros

boolean flagRecord=false; // drapeau pour activation enregistrement - true pour actif et false pour inactif

boolean debug=true; // true si message debogage, false sinon

// XXXXXXXXXXXXXXXXXXXXX  Fonction SETUP XXXXXXXXXXXXXXXXXXXXX

void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

        // ---- initialisation paramètres graphiques utilisés
        colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
        fill(0,0,255); // couleur remplissage RGB - noFill() si pas de remplissage
        stroke (0,0,0); // couleur pourtour RGB - noStroke() si pas de pourtour
        rectMode(CORNER); // origine rectangle : CORNER = coin sup gauche | CENTER : centre
        imageMode(CORNER); // origine image : CORNER = coin sup gauche | CENTER : centre
        ellipseMode(CENTER); // origine cercles / ellipses : CENTER : centre (autres : RADIUS, CORNERS, CORNER
        //strokeWeight(0); // largeur pourtour
        frameRate(fpsCapture);// Images par seconde - The default rate is 60 frames per second

        // --- initialisation fenêtre de base ---
        size(widthCapture, heightCapture); // ouvre une fenêtre xpixels  x ypixels
        background(0,0,0); // couleur fond fenetre

// --- initialisation des objets et fonctionnalités utilisées ---

        //======== Initialisation Objets GSVideo (capture et/ou lecture video =========

        // GSCapture(this, int requestWidth, int requestHeight, [int frameRate], [String sourceName], [String cameraName])
        cam1 = new GSCapture(this, widthCapture, heightCapture,"v4l2src","/dev/video0",fpsCapture); // Initialise objet GSCapture désignant webcam
        // largeur et hauteur doivent être compatible avec la webcam - typiquement 160x120 ou 320x240 ou 640x480...
        // Meilleurs résultats avec framerate webcam entre 20 et 30 et frameRate programme idem ou multiple plus grand (40 pour 20 par ex)
        // la liste des webcam installées sous Ubuntu (Gnu/Linux) est donnée par la commande : ls /dev/video*

        // cam1.play();  // démarre objet GSCapture = la webcam - version GSVideo avant 0.9
        cam1.start();  // démarre objet GSCapture = la webcam - version GSVideo après 0.9

        //------ initialisation objet GSMovieMaker pour enregistrement vidéo ---

          //GSMovieMaker(processing.core.PApplet parent, int requestWidth, int requestHeight, java.lang.String filename, int codecType, int codecQuality, int ifps)
          // IMPORTANT : le frameRate et ifps doivent être égaux !

          //----- initialisation de l'objet GSMovieMaker ---
          maVideo = new GSMovieMaker(this, widthCapture, heightCapture, cheminFichier+nomFichier, GSMovieMaker.MJPEG, GSMovieMaker.MEDIUM, fpsCapture);
          // ici enregistrement dans fichier.avi (cf nom fichier) avec encodage MJPEG et qualité MEDIUM


          // all quality settings are WORST, LOW,MEDIUM, HIGH and BEST

          // Available codecs are:
          // THEORA
          // XVID
          // X264
          // DIRAC
```

```
          // MJPEG
          // MJPEG2K
          // As for the file formats, the following are autodetected from the filename extension:
          // .ogg: OGG
          // .avi: Microsoft's AVI
          // .mov: Quicktime's MOV
          // .flv: Flash Video
          // .mkv: Matroska container
          // .mp4: MPEG-4
          // .3gp: 3GGP video
          // .mpg: MPEG-1
          // .mj2: Motion JPEG 2000
          // Please note that some of the codecs/containers might not work as expected, depending
          // on which gstreamer plugins are installed. Also, some codec/container combinations
          // don't seem to be compatible, for example THEORA+AVI or X264+OGG.

          //setQueueSize(int npre,int nenc)
          maVideo.setQueueSize(0, 10); // fixe le nombre de frame maxi en attente en pré-encodage
(java- 0 si pas de limite) et en post-encodage (gstreamer)

          // Sets the maximum size of the pre-encoding and encoding queues. When the encoding queue
is full, the frames start to be accumulated in the pre-encoding queue.
          // By setting the size of the pre-encoding queue to zero, it can grow arbitrarily large.



} // fin fonction Setup

// XXXXXXXXXXXXXXXXXXXXX Fonction Draw XXXXXXXXXXXXXXXXXXX

void  draw() { // fonction exécutée en boucle

// Code type capture GSVideo - utilisation possible aussi de captureEvent()

  if (cam1.available() == true) { // si une nouvelle frame est disponible sur la webcam
    cam1.read(); // acquisition d'un frame
    image(cam1, 0, 0); // affiche image
    //set(0, 0, cam); // affiche image - plus rapide

    imgDest=cam1.get(); // récupère l'image GS video dans Pimage

    //--- si l'enregistrement est activé ----
    if (flagRecord==true) {

        //--- trace cercle rouge dans coin sup droit
        fill(rouge);
        stroke(rouge);
        ellipse (widthCapture-30,30,20,20);

        //----- enregistrement des images ---
        imgDest.loadPixels(); // charge les pixels en mémoire
        maVideo.addFrame(imgDest.pixels); // ajoute les pixels de l'image à la vidéo - attention
ajoute les pixels et pas l'image...
        if (debug) println("Nombre de frames en attente : " + maVideo.getQueuedFrames());
        if (debug) println("Nombre de frame abandonnés : " + maVideo.getDroppedFrames());

    } // fin if flagRecord

  } // fin if available


        // while(true); // stoppe boucle draw

} // fin de la fonction draw()

// XXXXXXXXXXXXXXXXXXXXX Autres Fonctions XXXXXXXXXXXXXXXXXXXX

//------------ gestion évènement clavier ---------

void keyPressed() { // si une touche est appuyée


         if(key=='e') { // si touche ee enfoncee

           maVideo.start(); // lance l'objet GSMovieMaker
          println("Enregistrement démarré");
          flagRecord=true;
```

```
        }


        if(key=='s') { // si touche s enfoncee

          maVideo.finish(); // ferme l'objet MovieMaker
          println("Enregistrement stoppé");
          flagRecord=false;

        }


} //--- fin si touche enfoncee


//------------- Fonction d'arret de Processing ----

public void stop(){ // fonction d'arrêt de Processing

        cam1.delete(); // efface l'objet GScapture


        super.stop(); // obligatoire

} // fin fonction stop()


//XXXXXXXXXXXXXXXXXX Fin du programme XXXXXXXXXXXXXXXXX
```

*Fonctionnement, principe d'utilisation*

Le programme utilise :

- l'appui sur la touche e pour lancer l'enregistrement. Un rond rouge s'affiche sur l'écran.
- l'appui sur la touche s pour stopper l'enregistrement.

Le fichier pourra ensuite être lu en l'ouvrant directement dans le répertoire d'enregistrement.

**Test de la capture d'images à « haute-vitesse » (vitesse de capture à plus de 30fps et jusqu'à 120 fps)**

*Matériel*

Nécessite une webcam à capture rapide (supérieure à 30 fps) connectée à la GLAP-Box. Je recommande notamment la webcam Eye PS3 qui fonctionne « out of the box » avec la GLAP-Box et permet des captures jusqu'à 125 fps !

*Explications*

L'intérêt est :

- soit de visualiser des mouvements rapides
- soit d'obtenir des captures nettes d'objets rapides.

Le problème à de telles vitesses de capture est que le délai entre 2 images est seulement de 10ms. Il faut dès lors stocker les images en mémoire et ne les enregistrer que lorsque la capture est terminée. Attention : 10 secondes = 1000 images !

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
// Programme processing
// généré avec le générateur de code Processing
// du site www.mon-club-elec.fr
// par X. HINAULT - tous droits réservés

// Programme écrit le : 11/8/2011 - MAJ 04/2012. et 07/2012

// ------- Licence du code de ce programme : GPL v3-----


/////////////// Description du programme ////////////
// Utilise le clavier
// Utilise la librairie GSVideo de capture et lecture vidéo

// Utilise le clavier
// Utilise la librairie GSVideo de capture et lecture vidéo

/*
> ce programme affiche un flux vidéo capturé par la librairie GSVideo
> l'appui sur la touche 'e' enregistre le flux vidéo image par image dans des fichiers séparés
(Attention : efface l'enregistrement précédent)
> l'appui sur la touche 's' stoppe l'enregistrement

> la lecture des images se fait ensuite par ouverture du répertoire
> un logiciel tel que stopMotion permet la lecture enchaïnées des images

> le "truc" ici consiste à enregistrer les images dans un objet PGraphics = fenêtre Processing "off screen"

*/
// XXXXXXXXXXXXXXXXXXXXX ENTETE DECLARATIVE XXXXXXXXXXXXXXXXXXXXX

// inclusion des librairies utilisées

import codeanticode.gsvideo.*; // importe la librairie vidéo GSVideo qui implémente GStreamer pour Processing (compatible Linux)
// librairie comparable à la librairie native vidéo de Processing (qui implémente QuickTime..)- Voir Reference librairie Video Processing
// cette librairie doit être présente dans le répertoire modes/java/libraries du répertoire Processing (1-5)
// voir ici : http://gsvideo.sourceforge.net/
// et ici : http://codeanticode.wordpress.com/2011/05/16/gsvideo-09-release

// déclaration objets

PGraphics pg; // objet PGraphics pour créer une fenêtre Processing "offscreen"

GSCapture cam; // déclare un objet GSCapture représentant une webcam

PFont font;

PImage img;

String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/
String cheminRep=homePath+"Bureau/trans/captureImagesHighSpeed"+"_"+year()+"_"+month()+"_"+day()+"_"+hour()+"_"+minute()+"_"+second()+"/"; // chemin absolu du répertoire d'enregistrement/";
// déclaration variables globales

//------ déclaration des variables de couleur utiles ----
int jaune=color(255,255,0);
int vert=color(0,255,0);
int rouge=color(255,0,0);
int bleu=color(0,0,255);
int noir=color(0,0,0);
int blanc=color(255,255,255);
int bleuclair=color(0,255,255);
int violet=color(255,0,255);

int widthCapture=320; // largeur de la capture
int heightCapture=240; // hauteur de la capture
int fpsCapture=100; // fréquence de capture

boolean flagRecord=false; // drapeau pour activation enregistrement - true pour actif et false pour inactif

boolean debug=true; // true si message debogage, false sinon
```

```
long millis0Record=1;
long millis0LastImage=1;

int comptFrame=0;
String comptString="";

// XXXXXXXXXXXXXXXXXXXXX  Fonction SETUP XXXXXXXXXXXXXXXXXXXXX

void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

         pg = createGraphics(widthCapture, heightCapture, P2D); // initialise le PGraphic

        // ---- initialisation paramètres graphiques utilisés
        colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
        fill(0,0,255); // couleur remplissage RGB - noFill() si pas de remplissage
        stroke (0,0,0); // couleur pourtour RGB - noStroke() si pas de pourtour
        rectMode(CORNER); // origine rectangle : CORNER = coin sup gauche | CENTER : centre
        imageMode(CORNER); // origine image : CORNER = coin sup gauche | CENTER : centre
        ellipseMode(CENTER); // origine cercles / ellipses : CENTER : centre (autres : RADIUS,
CORNERS, CORNER
        //strokeWeight(0); // largeur pourtour
        frameRate(fpsCapture);// Images par seconde - The default rate is 60 frames per second

        // --- initialisation fenêtre de base ---
        //size(widthCapture, heightCapture); // ouvre une fenêtre xpixels  x ypixels
        background(0,0,0); // couleur fond fenetre

// --- initialisation des objets et fonctionnalités utilisées ---


        //------------- initialisation de la police texte - à mettre avant série ----
        font = loadFont("DejaVuSans-10.vlw"); // charge le fichier police dans l'objet police texte
        // ce fichier doit être présent dans un rép  dans le répertoire du programme
        // pour générer un fichier de police à partir des polices système aller dans Tools > create
Fonts
        // voir également http://processing.org/learning/text/

        pg.textFont(font, 15); // Initialise la police et sa taille (en pixels)

        // text("mon texte", 50, 50);


        //======== Initialisation Objets GSVideo (capture et/ou lecture video =========

        // GSCapture(this, int requestWidth, int requestHeight, [int frameRate], [String
sourceName], [String cameraName])
        cam = new GSCapture(this, widthCapture, heightCapture,"v4l2src","/dev/video0", fpsCapture);
// Initialise objet GSCapture désignant webcam
        // largeur et hauteur doivent être compatible avec la webcam - typiquement 160x120 ou
320x240 ou 640x480...


        cam.start();  // démarre objet GSCapture = la webcam - version GSVideo après 0.9


} // fin fonction Setup

// XXXXXXXXXXXXXXXXXXXXX Fonction Draw XXXXXXXXXXXXXXXXXXXX

void  draw() { // fonction exécutée en boucle

// Code type capture GSVideo - préférer utilisation de captureEvent()
  if (cam.available() == true) { // si une nouvelle frame est disponible
    cam.read(); // acquisition d'un frame
    pg.image(cam, 0, 0); // "affiche" image - ici enregistre image en mémoire-
    //pas d'affichage réel pour ne pas perdre de temps...
    //set(0, 0, cam); // plus rapide


    //img=cam.get(); // récupère l'image GS video dans Pimage

    //--- si l'enregistrement est activé ----
    if (flagRecord==true) {

        pg.beginDraw(); // debut de draw() PGraphic

        //--- trace cercle rouge dans coin sup droit
```

```
        pg.fill(rouge);
        pg.stroke(rouge);
        pg.ellipse (widthCapture-30,30,20,20);

        pg.fill(jaune);
        pg.text((millis()-millisORecord) + "ms | " + (millis()-millisOLastImage) + "ms  | " +
(1000.0/(millis()-millisOLastImage)) +"fps." , 10, 30);

        pg.endDraw(); // fin de draw() PGraphic

        //----- mise en forme comptString
        if (comptFrame<10) comptString="000"+comptFrame;
        if (comptFrame<100) comptString="000"+comptFrame;
        if (comptFrame<1000) comptString="00"+comptFrame;
        if (comptFrame<10000) comptString="0"+comptFrame;


        //----- enregistrement des images ---
       pg.save(cheminRep+"image-"+comptString+".jpg"); // enregistre une image numérotée dans le
rép indiqué
       // s'applique à l'aire courante de dessin - ici PGraphic
       //-- format disponible "tif", "tga", "jpg", "png"
       //-- saveImage va sauver avec toujours le meme nom de fichier...


       if (debug) println("image-"+comptFrame+".jpg");

       comptFrame=comptFrame+1;

    } // fin if flagRecord

        if (debug && flagRecord) background(rouge); else   background(noir);
        if (debug) println ("Délai  = " + (millis()-millisOLastImage) + "ms soit " + (1000/
(millis()-millisOLastImage)) +"fps.");
        millisOLastImage=millis();

  } // fin if available


        // while(true); // stoppe boucle draw


} // fin de la fonction draw()

// XXXXXXXXXXXXXXXXXXXXX Autres Fonctions XXXXXXXXXXXXXXXXXXXXX

//------------ gestion évènement clavier ---------

void keyPressed() { // si une touche est appuyée


         if(key=='e') { // si touche ee enfoncee

          println("Enregistrement démarré");
          millisORecord=millis()-2; // début enregistrement
          millisOLastImage=millis()-3; // début
          flagRecord=true;

        }


        if(key=='s') { // si touche s enfoncee

          println("Enregistrement stoppé");
          flagRecord=false;

        }


} //--- fin si touche enfoncee


//------------- Fonction d'arret de Processing ----

public void stop(){ // fonction d'arrêt de Processing

        cam.delete(); // efface l'objet GScapture
```

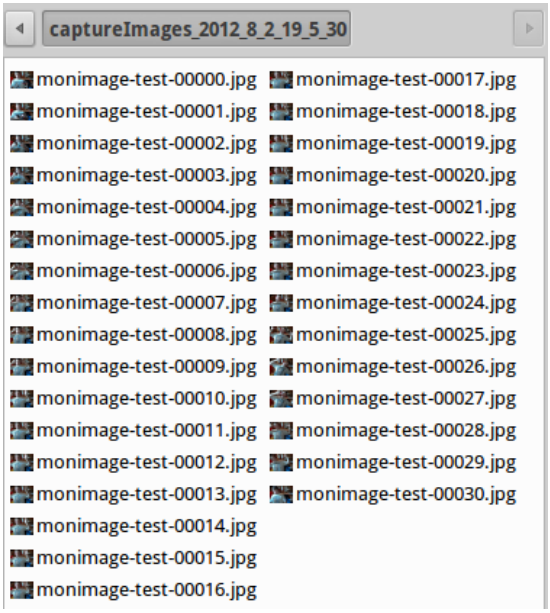
```
        super.stop(); // obligatoire
} // fin fonction stop()

//XXXXXXXXXXXXXXXXX Fin du programme XXXXXXXXXXXXXXXXX
```

Avant de lancer l'exécution, aller dans Tools > CreateFont pour ajouter la police DejaVuSans-10 qui est utilisée par ce programme.

*Fonctionnement, principe d'utilisation*

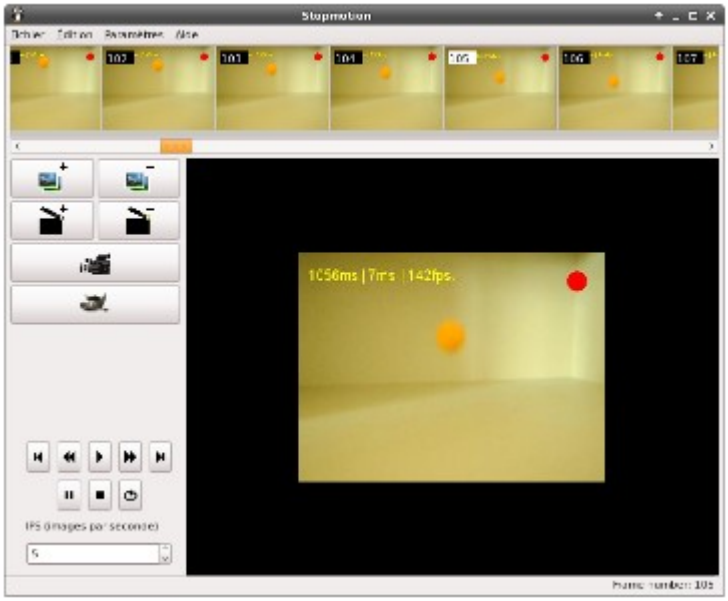
- l'appui sur la touche e pour lancer l'enregistrement. Un rond rouge s'affiche sur l'écran.
- l'appui sur la touche s pour stopper l'enregistrement. Noter que si on relance l'enregistrement, les images sont enregistrées à la suite.

Une fois la capture terminée, on obtient un répertoire horodaté qui contient les fichiers images numérotés :




Pour manipuler facilement ces images, il existe plusieurs logiciel. Sous Xubuntu et donc sur la GLAP-Box, il existe stopmotion, accessible depuis le menu XFCE > Multimédia > Stopmotion.

Avec ce logiciel, de cliquer sur le bouton « ajouter des images », puis d'ouvrir le répertoire des images, de toutes les sélectionner et de lancer la lecture en fixant la vitesse de défilement. Très pratique. Ce logiciel permet également d'exporter les images dans un fichier vidéo.




Sinon, pour des manipulations plus avancées, on pourra aussi utiliser kino sous Ubuntu (sur poste fixe).

La capture d'image à « haute vitesse » est très intéressante pour faire du « slow motion » = mouvement ralenti. Ceci est particulièrement intéressant lorsqu'on la couple à de la reconnaissance visuelle d'objet coloré, permettant la détection des nombreuses positions intermédiaires d'un objet en mouvement.

Exemple de suivi d'une balle à « haute vitesse » réalisé avec une GLAP-Box et ma librairie Javacvpro (OpenCV sous Processing).

## *Test de la reconnaissance visuelle et vision par ordinateur en temps réel*

Ces tests nécessitent une webcam connectée à la GLAP-Box sur un port USB.

### Test de la reconnaissance visuelle de visage

*Matériel*

Ce test nécessite une webcam connectée à la GLAP-Box.

*Explications*

Cette fonction utilise la librairie la librairie JavacvPro qui implémente OpenCV sous Processing. L'ensemble de la chaîne logicielle est déjà installée sur la GLAP-Box et le test suivant doit fonctionner immédiatement normalement.

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant (il s'agit de l'un des exemples de ma librairie JavacvPro) :

```
// Programme d'exemple de la librairie javacvPro
// par X. HINAULT - Mars 2012
// Tous droits réservés - Licence GPLv3

// Exemple fonction Cascade  - détection de visages

import codeanticode.gsvideo.*; // importe la librairie vidéo GSVideo qui implémente GStreamer pour Processing (compatible Linux)
// librairie comparable à la librairie native vidéo de Processing (qui implémente QuickTime..)- Voir Reference librairie Video Processing
// cette librairie doit être présente dans le répertoire modes/java/libraries du répertoire Processing (1-5)
// voir ici : http://gsvideo.sourceforge.net/

import monclubelec.javacvPro.*; // importe la librairie javacvPro
import java.awt.*; // pour classes Point , Rectangle..

PImage img;
Rectangle[] faceRect;

GSCapture cam; // déclare un objet GSCapture représentant une webcam
// L'objet GSCapture étend PImage - se comporte comme un conteneur des frames issues de la webcam

OpenCV opencv; // déclare un objet OpenCV principal

int widthCapture=320; // largeur image capture
int heightCapture=240; // hauteur image capture
int fpsCapture=30; // framerate de Capture

int millis0=0; // variable mémorisation millis()

void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

        //--- initialise fenêtre Processing
        size (widthCapture, heightCapture); // crée une fenêtre Processing de la 2xtaille du buffer principal OpenCV
        //size (img.width, img.height); // aalternative en se basant sur l'image d'origine
        frameRate(fpsCapture); // taux de rafraichissement de l'image

       //---- initialise la webcam ---
       cam = new GSCapture(this, widthCapture, heightCapture); // forme simplifiée
       //cam = new GSCapture(this, widthCapture, heightCapture,"v4l2src","/dev/video0", fpsCapture); // Initialise objet GSCapture désignant webcam - forme complète

        //--- initialise OpenCV ---
        opencv = new OpenCV(this); // initialise objet OpenCV à partir du parent This
        opencv.allocate(widthCapture, heightCapture); // initialise les buffers OpenCv à la taille de l'image

        cam.start();  // démarre objet GSCapture = la webcam

        //-- charge le fichier de description ---
        opencv.cascade("/usr/local/share/OpenCV/haarcascades/","haarcascade_frontalface_alt.xml"); // utilise chemin absolu Rép + nom fichier
        // supporte chemin absolu avec ou sans / en fin de chaine

}


void  draw() { // fonction exécutée en boucle

// Code capture GSVideo
```

```
  if (cam.available() == true) { // si une nouvelle frame est disponible sur la webcam

        //background(0);  // fond noir entre 2 images

        //------ gestion image webcam par GSCapture ------
        cam.read(); // acquisition d'un frame
        //image(cam1, 0, 0); // affiche image
        //set(0, 0, cam); // affiche image - plus rapide

        //------- gestion image par Opencv ----------

        img=cam.get(); // récupère l'image GS video dans Pimage

        millis0=millis(); // mémorise millis()
        opencv.copy(img); // charge l'image GSVideo dans le buffer openCV
        println("Durée chargement buffer OpenCV=" + (millis()-millis0)+"ms.");

        //--- affiche image de départ avant opération sur image ---
        image(img,0,0); // affiche le buffer principal OpenCV dans la fenêtre Processing

        //--- opérations sur image ---

        millis0=millis(); // mémorise millis()

        //faceRect = opencv.detect(true); // détection des visages avec messages debug
        faceRect = opencv.detect(3,true); // détection des visages avec coeff vitesse élevée et
messages debug

        opencv.drawRectDetect(true); // affiche les rectangles détectés avec messages debug

        println("Nombre de visages de face détectés =" + faceRect.length + ".");


        println("Durée traitement image par OpenCV=" + (millis()-millis0)+" ms.");

        //--- affiche image finale ---

        //image(opencv.getBuffer(),widthCapture,0); // affiche le buffer principal OpenCV dans la
fenêtre Processing

  } // fin if available


} // fin draw
```

Vous devez obtenir quelque chose comme çà à chaque fois qu'un visage apparaît dans le champ de la webcam :

### Test de suivi d'objet coloré

*Matériel*

Ce test nécessite une webcam connectée à la GLAP-Box.

Un objet coloré à suivre : ici une balle orangée.

*Explications*

La reconnaissance d'objet coloré repose par une « isolation » colorimétrique de l'objet à suivre au sein d'une image. Les fonctions de suivi et de recherche de forme sont assurées par la librairie JavacvPro qui implémente OpenCV sous Processing.

Ici, le programme, dans une visée didactique, affichera toutes les étapes de la détection dans 4 images séparées, afin de pouvoir améliorer les paramètres au besoin.

La balle utilisée pour ce test est une balle de ping pong Artengo (Decathlon) orange. La webcam testée est une Logitech C270 et une Hercules Dualpix Exchange.

*Le test*

Ouvrir Processing sur la GLAP-Box

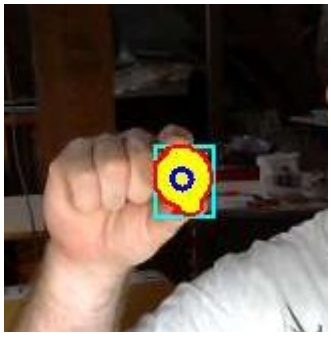
Copier/coller le code suivant :

```
// Programme d'exemple de la librairie javacvPro
// par X. HINAULT - Mars 2012
// Tous droits réservés - Licence GPLv3

// Exemple fonction selectBlobs - extraction de contour de formes binarisées

import codeanticode.gsvideo.*; // importe la librairie vidéo GSVideo qui implémente GStreamer pour
Processing (compatible Linux)
// librairie comparable à la librairie native vidéo de Processing (qui implémente QuickTime..)- Voi
r Reference librairie Video Processing
// cette librairie doit être présente dans le répertoire modes/java/libraries du répertoire Process
ing (1-5)
// voir ici : http://gsvideo.sourceforge.net/

import monclubelec.javacvPro.*; // importe la librairie javacvPro

PImage img;

Blob[] blobsArray=null; // tableau pour la détection des blobs (contour de forme)

GSCapture cam; // déclare un objet GSCapture représentant une webcam
// L'objet GSCapture étend PImage - se comporte comme un conteneur des frames issues de la webcam

OpenCV opencv; // déclare un objet OpenCV principal

int widthCapture=320; // largeur image capture
int heightCapture=240; // hauteur image capture
int fpsCapture=30; // framerate de Capture

int millis0=0; // variable mémorisation millis()

void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

        //--- initialise fenêtre Processing
        size (widthCapture*2, heightCapture*2); // crée une fenêtre Processing de la 2xtaille du
buffer principal OpenCV
        //size (img.width, img.height); // aalternative en se basant sur l'image d'origine
        frameRate(fpsCapture); // taux de rafraichissement de l'image

       //---- initialise la webcam ---
       cam = new GSCapture(this, widthCapture, heightCapture); // forme simplifiée
```

```
      //cam = new GSCapture(this, widthCapture, heightCapture,"v4l2src","/dev/video0",
fpsCapture); // Initialise objet GSCapture désignant webcam - forme complète

        //--- initialise OpenCV ---
        opencv = new OpenCV(this); // initialise objet OpenCV à partir du parent This
        opencv.allocate(widthCapture, heightCapture); // initialise les buffers OpenCv à la taille
de l'image

        cam.start();  // démarre objet GSCapture = la webcam

}


void  draw() { // fonction exécutée en boucle

// Code capture GSVideo

  if (cam.available() == true) { // si une nouvelle frame est disponible sur la webcam

        background(0);  // fond noir entre 2 images

        //------ gestion image webcam par GSCapture ------
        cam.read(); // acquisition d'un frame
        //image(cam1, 0, 0); // affiche image
        //set(0, 0, cam); // affiche image - plus rapide

        //------- gestion image par Opencv ----------

        img=cam.get(); // récupère l'image GS video dans Pimage

        millis0=millis(); // mémorise millis()
        opencv.copy(img); // autre possibilité - charge directement l'image GSVideo dans le buffer
openCV
        println("Durée chargement buffer OpenCV=" + (millis()-millis0)+"ms.");

        //--- affiche image de départ avant opération sur image ---
        image(opencv.getBuffer(),0,0); // affiche le buffer principal OpenCV dans la fenêtre
Processing

        //--- opérations sur image ---

        millis0=millis(); // mémorise millis()

        //--- application du filtre mixerRGBGray()
        //opencv.mixerRGBGray(); // applique mixeur RGBGray sur le buffer principal OpenCV avec
paramètres par défaut (1.0, 1.5,-2.0
        opencv.mixerRGBGray(1.0,1.5, -2.0); // mixerRGBGray appliqué sur objet IplImage avec
paramètres - ici détection couleur orangée
        image(opencv.getBuffer(),opencv.width(),0); // affiche le buffer principal OpenCV dans la
fenêtre Processing

        //--- application d'un seuillage binaire ---
        opencv.threshold(0.8, "BINARY"); // seuillage binaire pour éliminer le fond
        image(opencv.getBuffer(),0,opencv.height()); // affiche le buffer principal OpenCV dans la
fenêtre Processing

        //-- détection de blobs ---
        //blobs=opencv.blobs(true); // blobs javacvPro avec paramètres par défaut +/- debug

        //blobsArray=blobs(minArea,maxArea,maxBlob, findHoles, maxVertices, debug);
        blobsArray = opencv.blobs(opencv.area()/4000, opencv.area()/2, 20, true, 1000, true ); //
blobs javacvPro +/- debug
        // la fonction renvoie un tableau de Blob
        // chaque Blob est caractérisé par son aire, son centre, le rectangle entourant, le tableau
des points du contour
        // pour info : la fonction blobs mémorise en interne un tableau des séquences de points
détectés (jusqu'à l'appel suivant de la fonction blobs() )
        // pour utilisation avec les fonctions avancées convexHull et detectDefect par exemple

        //-- dessin du rectangle autour du tracé de la forme avant sélection --
        //opencv.drawRectBlobs (blobsArray, xRef, yRef, scale, colorStroke, strokeWeight, fill,
colorFill)
        opencv.drawRectBlobs(blobsArray,0,opencv.height(),1, color(255,0,255),1,false,0); // trace
le rectangle avec les paramètres

        //--- sélection des Blob cohérents ---
```

```
        blobsArray=opencv.selectBallBlobs(blobsArray); // sélectionne uniquement les Blobs pouvant
correspondre à 1 balle

        //-- réaffichage image de départ --
        image(img,opencv.width(), opencv.height());

        //-- dessin du rectangle autour du tracé de la forme --
        //opencv.drawRectBlobs (blobsArray, xRef, yRef, scale, colorStroke, strokeWeight, fill,
colorFill)
        // -- toutes ces formes sont possibles
        //opencv.drawRectBlobs(blobsArray); // trace le rectangle avec les paramètres par défaut
        opencv.drawRectBlobs(blobsArray,opencv.width(),opencv.height(),1); // trace rectangle en se
basant sur point référence et avec les paramètres
        //opencv.drawRectBlobs(blobsArray,opencv.width(),opencv.height(),1, color(255,0,255), 2,
false, 0); // trace rectangle avec les paramètres

        //-- dessin du pourtour du blob sur l'image de départ --
        //opencv.drawBlobs (blobsArray, xRef, yRef, scale, radius, colorStroke, strokeWeight, fill,
colorFill, mode);
        //-- toutes ces formes sont possibles
        //opencv.drawBlobs(blobsArray); // trace les formes du tableau de Blobs avec paramètre par
défaut
        opencv.drawBlobs(blobsArray,opencv.width(),opencv.height(),1 ); // trace les formes du
tableau de Blobs en se basant sur point référence + paramètre par défaut
        //opencv.drawBlobs(blobsArray,opencv.width(),opencv.height(),1,5, color(0,0,255), 1, false,
0, 0); // trace pourtour en cercles bleus

        //-- dessin du centre du blob sur le tracé de la forme --
        // opencv.drawCentroidBlobs (blobsArray, xRef, yRef, scale, radius, colorStroke,
strokeWeight, fill, colorFill);
        // -- toutes ces formes sont possibles
        //opencv.drawCentroidBlobs (blobsArray); // trace le centre des Blob avec les paramètres
par défaut
        opencv.drawCentroidBlobs (blobsArray,opencv.width(),opencv.height(),1); // trace le centre
des Blob en se basant sur point référence + échelle et avec les paramètres par défaut
        //opencv.drawCentroidBlobs (blobsArray,opencv.width(),opencv.height(),1,10); // trace le
centre des Blob en se basant sur point référence + échelle + radius et avec les paramètres par
défaut
        //opencv.drawCentroidBlobs (blobsArray,opencv.width(),opencv.height(),1,5, color(0,0,255),
1, false, 0); // trace le centre des Blob en fonction paramètres


        println("Durée traitement image par OpenCV=" + (millis()-millis0)+" ms.");

        //--- affiche image finale ---

        //image(opencv.getBuffer(),widthCapture,0); // affiche le buffer principal OpenCV dans la
fenêtre Processing

  } // fin if available

} // fin draw
```

Vous devriez obtenir quelque chose qui ressemble à çà :



Ce programme fait 2 choses en fait :

- filtrage colorimétrique d'une part pour isoler la couleur voulue,
- filtrage « géométrique » des formes de façon à isoler la forme ronde.

Couplé à une carte Arduino, ces fonctions de suivi d'objet colorés permettront d'obtenir des mouvements de « tracking » d'objet par une webcam ou un robot, ou encore d'assurer la détection de la position d'un objet à prendre avec un bras robotisé à servomoteurs.

#### Suivi d'objet coloré à « haute vitesse » de capture

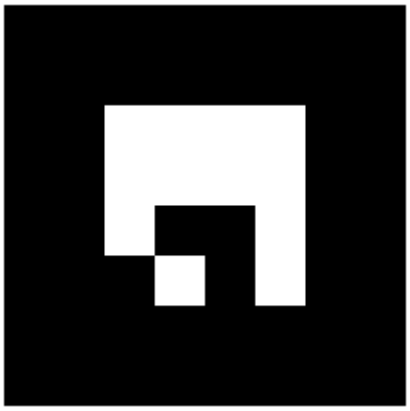
Exemple de suivi d'une balle à « haute vitesse » réalisé avec une GLAP-Box et ma librairie Javacvpro (OpenCV sous Processing).

#### Test de la reconnaissance de marker ARToolkit

*Matériel*

Ce test nécessite une webcam connectée à la GLAP-Box.

Vous devez également disposer d'un marker à reconnaître.



Voir ici comment en obtenir un :

http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.OutilsProcessingARToolkitAvecProcessing

*Explications*

Cette fonction est basée sur la librairie Nyar4Psg qui implémente ARToolkit sous Processing. La librairie JavacvPro intègre des fonctions qui permettent de faciliter la détection des markers.

*Le test*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
// Programme processing
// généré avec le générateur de code Processing
// du site www.mon-club-elec.fr
// par X. HINAULT - tous droits réservés

// Programme écrit le : 29/2/2012.

// ------- Licence du code de ce programme : GPL v3-----
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License,
//  or any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

/////////////// Description du programme ////////////
// Utilise la librairie GSVideo de capture et lecture vidéo
// Utilise la librairie javacvPro de traitement d'image et reconnaissance visuelle
// Utilise la librairie nyar4psg qui implémente ARToolkit dans Processing

/*
Détection du centre et du type d'un marker ARToolkit
*/

// XXXXXXXXXXXXXXXXXXXXX ENTETE DECLARATIVE XXXXXXXXXXXXXXXXXXXXX

// inclusion des librairies utilisées

import codeanticode.gsvideo.*; // importe la librairie vidéo GSVideo qui implémente GStreamer pour
Processing (compatible Linux)
// librairie comparable à la librairie native vidéo de Processing (qui implémente QuickTime..)-
Voir Reference librairie Video Processing
// cette librairie doit être présente dans le répertoire modes/java/libraries du répertoire
Processing (1-5)
// voir ici : http://gsvideo.sourceforge.net/
// et ici : http://codeanticode.wordpress.com/2011/05/16/gsvideo-09-release

import monclubelec.javacvPro.*; // importe la librairie javacvPro qui implémente le traitement
d'image avancé et la reconnaissance visuelle pour Processing
// cette librairie se base sur les fonctions java de la librairie javacv par Samuel Audet :
http://code.google.com/p/javacv/
// javacv implémente en Java les centaines de fonctions natives de la librairie OpenCV (2500
algorithmes) !
// la librairie javacvPro doit être présente dans le répertoire modes/java/libraries du répertoire
Processing (1-5)
// dispo ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_javacvPro/pmwiki.php
// nécessite également que la librairie native OpenCV 2.3.1 soit installée sur votre ordinateur
// NB : compatibilité avec la plupart des fonctions de la librairie OpenCV pour Processing :
http://ubaa.net/shared/processing/opencv/

import java.awt.*; // Classe Point, Rectangle , etc...

import jp.nyatla.nyar4psg.*; // La librairie NyARToolkit Processing library = ARToolKit pour
Processing
// Cette librairie permet de détecter des pattern ou marker dans une image
// et d'analyser leur transformation de perspective
// nyAR4psg est à télécharger ici : http://sourceforge.jp/projects/nyartoolkit/releases/
// et à mettre dans le répertoire des librairies
// à noter : javacvPro intègre des fonctions utilisables directement avec nyar4psg
```

```
// déclaration objets

PImage imgSrc, imgDest; // déclare un/des objets PImage (conteneur d'image Processing)

GSCapture cam; // déclare un objet GSCapture représentant une webcam
// L'objet GSCapture étend PImage - se comporte comme un conteneur des frames issues de la webcam

OpenCV opencv; // déclare un objet OpenCV principal

PMatrix3D syst3D; // déclare une matrice 4x4 représentant un système de coordonnées 3D..

MultiMarker nya; // déclaration de l'objet principal pour reconnaissance des markers - nya pour
nyArtoolkit

// déclaration variables globales


//------ déclaration des variables de couleur utiles ----
int jaune=color(255,255,0);
int vert=color(0,255,0);
int rouge=color(255,0,0);
int bleu=color(0,0,255);
int noir=color(0,0,0);
int blanc=color(255,255,255);
int bleuclair=color(0,255,255);
int violet=color(255,0,255);


// variable pour la taille de la capture video
int widthCapture=320*2; // largeur capture
int heightCapture=240*2; // hauteur capture
int fpsCapture=15; // framerate (image/secondes) pour la capture video

// NB : pour la détection des markers, on peut utiliser une grande résolution facilement.
// Le résultat sera quand même rapide... et la précision plus élevée.

//----- variables pour calibration webcam -----
// float ouvertureX=19.43; // ouverture largeur en degres - Hercules DualPix Exchange
 float ouvertureX=22.53; // ouverture largeur en degres -  Logitech C270

// calculé avec tan angle= largeur réelle / 2 * distance camera
// exemple Logitech C270 : 1/2 largeur réelle = 83cm  distance = 200 cm
// d'où tan angle = 0.415 et d'où angle = 22.53 deg

//--------- variables pour reconnaissance des Markers avec nyARToolkit  ---------------------

String homePath=System.getProperty("user.home")+"/"; // définit le chemin utilisateur /home/user/

//----- chemin absolu fichier de paramètres de distorsion de la camera ----
//String camParamPath = "/home/hinault/Téléchargements/librairies_processing/nyar4psg-
1.1.6/data/camera_para.dat";
String camParamPath = homePath+"Téléchargements/processing-
1.5.1/modes/java/libraries/NyAR4psg/data/camera_para.dat";
// utilise le fichier par défaut - donne résultat satisfaisant

//----- chemin absolu fichiers de description des patterns ou markers ----
String patternPath = homePath+"Téléchargements/patternMaker/examples/ARToolKit_Patterns";
// à télécharger ici : http://www.cs.utah.edu/gdc/projects/augmentedreality/

//--- taille de l'image à utiliser pour la détection = plus petite pour plus rapide ---
int widthAR= widthCapture;
int heightAR=heightCapture;

int numMarkers = 8; // le nombre de pattern ou markers différents à utiliser

Marker[] markersArray = new Marker[numMarkers]; // tableau pour stockage des paramètres des markers
détectés avec ARToolkit - classe javacvPro !

float realWidthMarker=50; // taille réelle du marker utilisé en mmm - on aura un correspondance 1
mm = 1 pixel ou cran dans le repère 3D du marker

//String[] nameMarkers= new String[numMarkers]; // pour mémoriser le nom des marker


// XXXXXXXXXXXXXXXXXXXXX  Fonction SETUP XXXXXXXXXXXXXXXXXXXXX
```

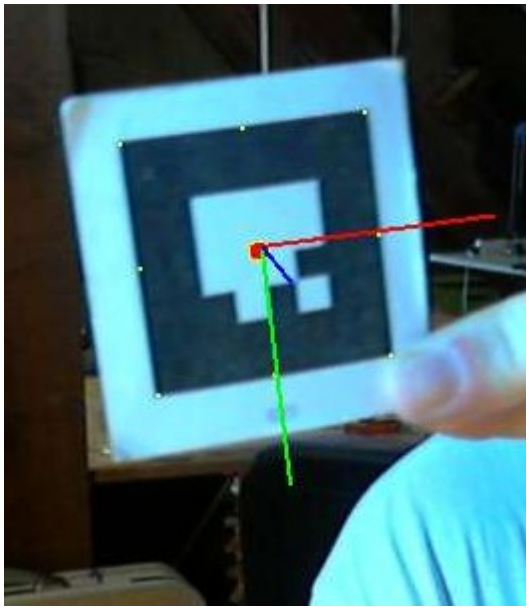
```
void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

    // ---- initialisation paramètres graphiques utilisés
    colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
    fill(0,0,255); // couleur remplissage RGB - noFill() si pas de remplissage
    stroke (0,0,255); // couleur pourtour RGB - noStroke() si pas de pourtour
    rectMode(CORNER); // origine rectangle : CORNER = coin sup gauche | CENTER : centre
    imageMode(CORNER); // origine image : CORNER = coin sup gauche | CENTER : centre
    ellipseMode(CENTER); // origine cercles / ellipses : CENTER : centre (autres : RADIUS,
CORNERS, CORNER
    //strokeWeight(0); // largeur pourtour
    frameRate(30);// Images par seconde - The default rate is 60 frames per second

    // --- initialisation fenêtre de base ---
    size(widthCapture,heightCapture,P3D); // ouvre une fenêtre xpixels  x ypixels - 3D active
    background(0,0,0); // couleur fond fenetre



// --- initialisation des objets et fonctionnalités utilisées ---

    //======== Initialisation Objets GSVideo (capture et/ou lecture video =========

    // GSCapture(this, int requestWidth, int requestHeight, [int frameRate], [String sourceName],
[String cameraName])
    // cam1 = new GSCapture(this, widthCapture, heightCapture,fpsCapture,"v4l2src","/dev/video0");
// Initialise objet GSCapture désignant webcam - avant GSVideo 1.0
    cam = new GSCapture(this, widthCapture, heightCapture,"v4l2src","/dev/video0", fpsCapture); //
Initialise objet GSCapture désignant webcam - depuis GSVideo 1.0
    // largeur et hauteur doivent être compatible avec la webcam - typiquement 160x120 ou 320x240
ou 640x480...
    // Meilleurs résultats avec framerate webcam entre 20 et 30 et frameRate programme idem ou
multiple plus grand (40 pour 20 par ex)
    // la liste des webcam installées sous Ubuntu (Gnu/Linux) est donnée par la commande : ls
/dev/video*

    // cam1.play();  // démarre objet GSCapture = la webcam - version GSVideo avant 0.9
    cam.start();  // démarre objet GSCapture = la webcam - version GSVideo après 0.9

    //======== Initialisation Objets OpenCV (librairie javacvPro : traitement d'image et
reconnaissance visuelle) =========

    opencv = new OpenCV(this); // initialise objet OpenCV à partir du parent This
        opencv.allocate(widthCapture,heightCapture); // crée les buffers image de la taille voulue


     //=========== initialisation détection des markers ========================

     // création d'un objet MultiMarker avec résolution voulue, les paramètres caméra et le système
de coordonnées voulu
        nya = new MultiMarker(this, widthAR, heightAR, camParamPath,
NyAR4PsgConfig.CONFIG_DEFAULT);

   // fixe le nombre de fois qu'un marqueur ne doit plus etre détecté pour ne plus l'afficher.
     //Par défaut = 10. Mettre à 1 pour visualisation immédiate
      nya.setLostDelay(1);

      // fixe le niveau de seuil de détection à utiliser. Valeur possible entre 0 et 255. Mettre -1
(=THLESHOLD_AUTO) pour seuil automatique - respecter la "faute"
        nya.setThreshold(MultiMarker.THLESHOLD_AUTO);

        // fixe le niveau de seuil de confiance (= probabilité de correspondance) à utiliser pour
la reconnaissance des markers. Valeur possible entre 0 et 1.
        // Valeur par défaut = 0.51 (=.DEFAULT_CF_THRESHOLD). Plus le seuil est élevé et plus la
détection est rigoureuse.
        nya.setConfidenceThreshold(MultiMarker.DEFAULT_CF_THRESHOLD);
        //nya.setConfidenceThreshold(0.8); // sélection exigeante

        //-- chargement des fichiers de description des patterns

        //------- création des objets Marker individuel du tableau de markers (javacvPro)
        // et initialisation des propriétés communes
        for (int i=0; i<markersArray.length; i++) {
          markersArray[i]=new Marker();
          markersArray[i].realWidth=realWidthMarker;
        } // fin for
```

```
        //-------- initialisation du tableau d'objets marker (javacvPro)
        // chaque objet Marker de la librairie javacvPro contient les paramètres attachés au marker
nyar correspondant
        // son nom = le fichier de description *.patt
        markersArray[0].name="4x4_99.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[1].name="4x4_50.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[2].name="4x4_83.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[3].name="4x4_23.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[4].name="4x4_68.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[5].name="4x4_76.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[6].name="4x4_55.patt"; // mémorise le nom du fichier du marker voulu
        markersArray[7].name="4x4_34.patt"; // mémorise le nom du fichier du marker voulu
        // etc...

        //---- chargement des markers dans le MultiMarker (nyartoolkit)---
        for (int i=0; i<markersArray.length; i++) {
          nya.addARMarker(patternPath + "/" + markersArray[i].name, markersArray[i].realWidth); //
ajoute le fichier de description à l'objet principal de détection AR - bordure 25% et 16x16 par
défaut
          println ("Fichier chargé : " + markersArray[i].name);
        } // fin for


} // fin fonction Setup

// XXXXXXXXXXXXXXXXXXXXX Fonction Draw XXXXXXXXXXXXXXXXXXX

void  draw() { // fonction exécutée en boucle


// Code type capture GSVideo

  if (cam.available() == true) { // si une nouvelle frame est disponible sur la webcam

    background(0);  // fond noir entre 2 images

    //------ gestion image webcam par GSCapture ------
    cam.read(); // acquisition d'un frame
    image(cam, 0, 0); // affiche image
    //set(0, 0, cam); // affiche image - plus rapide


    //------- gestion détection et dessin 2D des Markers ----------

    nya.detect(cam); // detection des markers dans l'image à la résolution voulue - nyar4psg
    // l'image passée en paramètre doit avoir la même résolution que ce qui a été défini à
l'initialisation du constructeur

    println("seuil de binarisation actuel = " + nya.getCurrentThreshold()); // affiche le niveau
courant du seuillage binaire

    opencv.updateMarkers(nya, markersArray, false); // met à jour le tableau des paramètres 2D des
markers - javacvPro

    //--- dessin 2D ---

   //public void draw2DMarkers(MultiMarker nyaIn, Marker[] markersIn, int xRefIn, int yRefIn, float
scaleIn, int radius, int colorStrokeIn, int strokeWeightIn, boolean fillIn, int colorFillIn,
boolean debugIn)
   opencv.draw2DMarkers(nya, markersArray); // trace les markers - javacvPro

   opencv.distanceMarkers (nya, markersArray, ouvertureX, widthCapture, false);  // calcule et
mémorise la distance réelle des Markers à la caméra - javacvPro

   // accéder aux caractéristiques 2D des markers ? - le centre, la rotation du marker sur lui-
meme...
   // sélectionner un Marker voulu...


/* --- première possibilité --
   if (nya.isExistMarker(3)) { // si le marker indice 3 est détecté

     println("Le marker " + markersArray[3].name + " est détecté !"); // le nom du fichier de
description du marker

   }
*/
```

```
/*---- 2ème possibilité ---

   if (opencv.isExistMarker(nya, markersArray, "4x4_23.patt")) { // détecte si marker détecté à
partir de son nom

          println("Le marker " + "4x4_23.patt"+ " est détecté !"); // le nom du fichier de
description du marker

   }

*/


// ---- 3ème possibilité ----
   if (opencv.isExistMarker(nya, markersArray,23)) { // détecte si marker détecté à partir de son
numero (pas l'indice)

         Marker selectedMarker = opencv.selectMarker(markersArray,23); // sélectionne le marker
voulu

          println("Le marker " + selectedMarker.name + " est détecté !"); // le nom du fichier de
description du marker

          println("Le centre est en x=" + selectedMarker.center2D.x + " | y="+
selectedMarker.center2D.y); // affiche les coordonnées du centre


          opencv.drawCircle (
                    selectedMarker.center2D, // le centre du cercle à tracer
                        0, 0, // les coordonnées du cercle à tracer
                        1, // l'échelle à utiliser
                        10, // rayon à utiliser
                        jaune,1, // couleur et épaisseur du pourtour du cercle
                        true, rouge, // drapeau de remplissage et couleur de remplissage
                        false // drapeau d'affichage des messages
                                );


   }



/* // pour mémoire : les champs de l'objet Marker

     public String name       ="" ; // le nom du fichier de description du marker

     public float realX       = (float) 0.0; // abscisse réelle au sol de l'espace d'évolution
décrit par les markers
     public float realY       = (float) 0.0; // ordonnée réel au sol de l'espace d'évolution
décrit par les markers

     public float realWidth       = (float) 0.0; // largeur réelle du Marker

     public float width2D     = (float) 0.0; // largeur 2D du Marker telle que affichée sur
l'image webcam
     public float height2D        = (float) 0.0; // hauteur 2D du Marker telle que affichée sur
l'image webcam

     //--- pour réalité augmentée --
     public float width3D     = (float) 1000; // largeur 3D du Marker telle que affichée sur
l'image webcam
     public float height3D        = (float) 1000; // hauteur 3D du Marker telle que affichée sur
l'image webcam
     public float depth3D     = (float) 10; // profondeur 3D du Marker telle que affichée sur
l'image webcam

     public float distance        = (float) 0.0; // distance du marker à la webcam (calculée)

     public float angleAxeY       = (float) 0.0; // angle de rotation dans l'axe Y en degrés


     public Point upCenter2D = new Point(); // milieu bord sup 2D du marqueur
     public Point downCenter2D = new Point(); // milieu bord inf 2D du marqueur
     public Point leftCenter2D = new Point(); // milieu bord gauche 2D du marqueur
     public Point rightCenter2D = new Point(); // milieu bord droit 2D du marqueur
```

```
     public Point center2D = new Point(); // centre 2D du marqueur

     public Point[] corners2D = new Point[4]; // coins 2D du marqueur
*/


    //---- dessin 3D ---
    //--- opération 3D à faire APRES détection 2D

    //--- affiche repère 3D centré sur l'écran ---
    //translate(width/2,height/2,0); // translation du repère 3D courant de pour centrage au centre
de la fenêtre camera
    //opencv.drawCurrentSyst3D(50, color(255,0,0), color(0,255,0), color(0,0,255),2); // affiche le
repère 0xyz courant avec taille et épaisseur voulus

    //void draw3DMarkers(MultiMarker nyaIn, Marker[] markersIn, int widthBoxIn, int heightBoxIn,
int depthBoxIn, boolean strokeIn, int colorStrokeIn, int strokeWeightIn, boolean fillIn, int
colorFillIn, boolean debugIn)
    opencv.draw3DMarkers(nya, markersArray,0, 0,10, false, 0, 0, false, color(255,0,0), true); //
si width et height=0, utilise propriétés 3D de chaque marker

  } // fin if available


    // while(true); // stoppe boucle draw

} // fin de la fonction draw()

// XXXXXXXXXXXXXXXXXXXXXX Autres Fonctions XXXXXXXXXXXXXXXXXXXXXX


//------------- Fonction d'arret de Processing ----

public void stop(){ // fonction d'arrêt de Processing

    cam.delete(); // efface l'objet GScapture

    super.stop(); // obligatoire

} // fin fonction stop()
```

Vous devriez obtenir quelque chose comme çà (remarquer la détection 3D du marker) :



Ces markers permettront des fonctions avancées de reconnaissance visuelle, de calcul de distance et de mesure d'angle par rapport à un objectif de visée. Ces markers présentent plusieurs avantages décisifs :

- leur détection est assez tolérante aux variations de luminosité.

- ce sont des outils très pratiques dont la reconnaissance est rapide comparativement à d'autres techniques.
- La détection est discriminante permettant de reconnaître spécifiquement un marker parmi plusieurs dizaines voire centaine avec une bonne précision : ceci pourra être à la base de comportements évolués...
- la détection est sensible à la rotation et à l'orientation du marker dans l'espace permettant des analyses avancées 3D.

Cool non ?

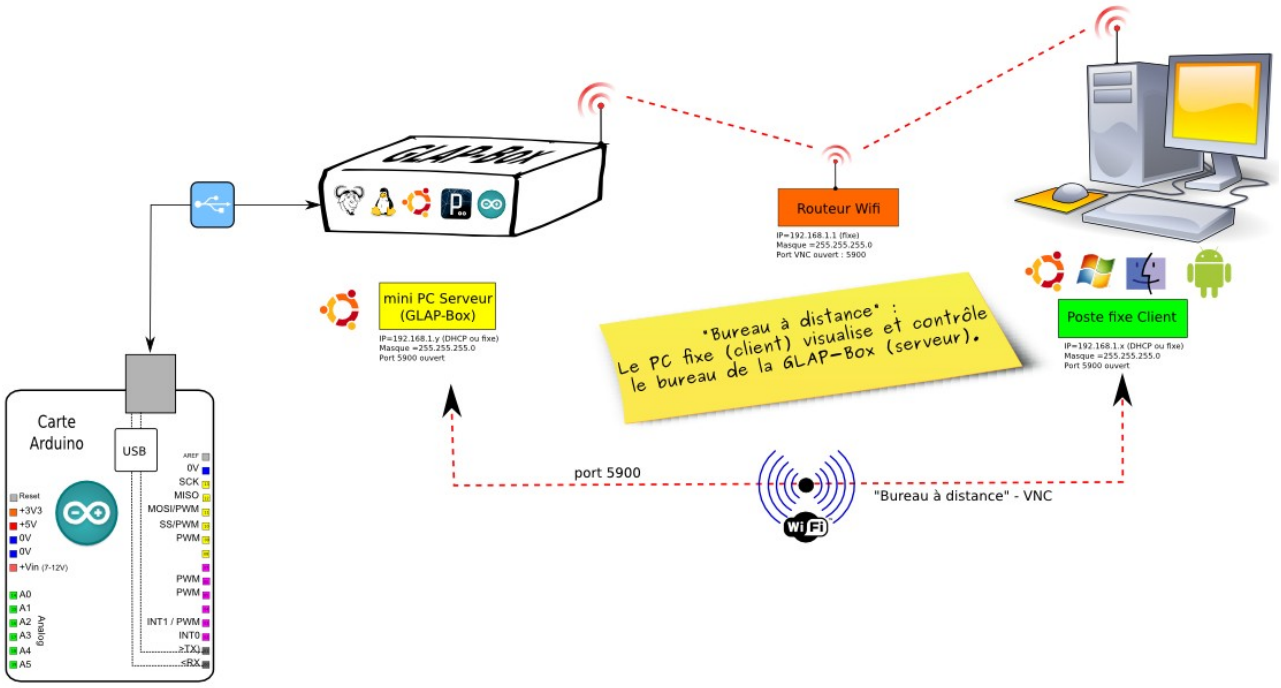
La reconnaissance des Markers ARToolkit est un domaine où l'utilisation de la GLAP-Box trouve tout son sens : leur utilisation nécessite l'installation de plusieurs librairies, de fichier de description, etc... Avec la GLAP-Box, vous avez du « clé en main » !

## *Test de communication série avec la carte Arduino à partir de Processing*

### Test simple miroir

*Matériel*

Ce test nécessite une carte Arduino connectée à la GLAP-Box.


Routeur Wifi
mini PC Serveur (GLAP-Box)
Poste fixe Client
"Bureau à distance" : Le PC fixe (client) visualise et contrôle le bureau de la GLAP-Box (serveur).
Carte Arduino
USB
port 5900
"Bureau à distance" - VNC
WiFi

*Explications*

Ici, nous allons simplement tester, par accès distant au bureau de la GLAP-Box depuis le poste fixe, que la comunication se fait bien entre un programme Processing et la carte Arduino connectée à la GLAP-Box.

Le code Arduino est un simple « miroir » qui va renvoyer sur le port série la chaîne reçue sur le port Série.

Le code Processing va consister à permettre la saisie d'une chaîne de caractère vers Arduino à l'aide d'un champ de saisie. La chaîne reçue sur le port série par Processing sera affichée dans la console.

*Le test côté Arduino*

Ouvrir Arduino sur la GLAP-Box

Copier/coller le code suivant :

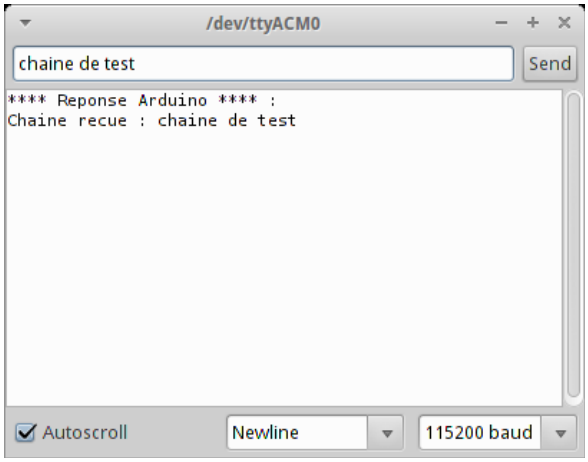
```
// --- Programme Arduino ---

// Auteur du Programme : X. HINAULT - Tous droits réservés
// Programme écrit le : 02/02/2011.

// ------- Licence du code de ce programme -----

// ////////////////////  PRESENTATION DU PROGRAMME ////////////////////

// -------- Que fait ce programme ? ---------
 /* Le programme reçoit une chaîne de caractère se terminant
 par un saut de ligne (depuis une interface Processing)
 sur le port série USB. Cette chaîne reçue est renvoyée vers Processing
 et peut etre affichée dans la console Processing.
 Ce programme utilise la nouvelle classe String du langage Arduino.  */

// --- Fonctionnalités utilisées ---

// Utilise la connexion série vers le PC

// -------- Circuit à réaliser ---------

// La connexion série vers le PC utilise les broches 0 et 1 (via le câble USB)


// ////////////////////////////// 1. Entête déclarative //////////////////////
// A ce niveau sont déclarées les librairies incluses, les constantes, les variables, les objets ut
iles...

// --- Déclaration des variables globales ---

int octetReception=0; // variable de stockage des valeurs reçues sur le port Série (ASCII)
char caractereRecu=0; // variable pour stockage caractère recu
int compt=0; // variable comptage caractères reçus

String chaineReception=""; // déclare un objet String vide pour reception chaine


// ////////////////////////// 2. FONCTION SETUP = Code d'initialisation ////////////////////////
// La fonction setup() est exécutée en premier et 1 seule fois, au démarrage du programme

void setup()   { // debut de la fonction setup()

// --- ici instructions à exécuter 1 seule fois au démarrage du programme ---

// ------- Initialisation fonctionnalités utilisées -------

Serial.begin(115200); // initialise connexion série à 115200 bauds
// IMPORTANT : régler le terminal côté PC avec la même valeur de transmission


} // fin de la fonction setup()
// ******************************************************************************

////////////////////////////////// 3. FONCTION LOOP = Boucle sans fin = coeur du programme ////////
//////////
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino est sous tension

void loop(){ // debut de la fonction loop()


//---- code type réception valeur sur le port série ---

while (Serial.available()>0) { // tant qu'un octet en réception

        octetReception=Serial.read(); // Lit le 1er octet reçu et le met dans la variable
        compt=compt+1;

        //Serial.println("Ascii Caractere "+ String(compt) +" = "+ String(octetReception)); //
affiche code ASCII Octet reçu
```

```
        if (octetReception==10) { // si Octet reçu est le saut de ligne

          //Serial.println("Saut de ligne recu");
          Serial.println("**** Reponse Arduino **** :");
          Serial.println ("Chaine recue : "+chaineReception); // affiche la chaine recue
          chaineReception=""; //RAZ le String de réception
          compt=0; // RAZ compteur
          delay(100); // pause
          break; // sort de la boucle while

        }
        else { // si le caractère reçu n'est pas un saut de ligne

           caractereRecu=char(octetReception); // convertit l'octet reçu en caractère
           chaineReception=chaineReception+caractereRecu; // ajoute le caratère au String
        }

} // fin tant que  octet réception

//----- une fois que le saut de ligne est reçu, on sort du While et on se positionne ici



} // fin de la fonction loop() - le programme recommence au début de la fonction loop sans fin
// ********************************************************************************


// ////////////////////////// Fin du programme ///////////////////
```

Programmer la carte Arduino.

Une fois la carte Arduino programmée, pour vérifier le bon fonctionnement du programme :

- ouvrir le Terminal Série, fixer le débit à 115200 et choisir l'option Newline (un saut de ligne sera ajouté après la chaîne)
- Si tout est ok, toute chaîne saisie dans le champ de saisie du Terminal doit s'afficher dans le Terminal.




- Une fois que vous êtes sûr que çà marche bien, **fermer la fenêtre Terminal.** Ne pas oublier de le faire ! Sinon, le port série de la carte Arduino est déjà occupé et Processing ne pourra pas s'y connecter...

Vous voyez ici un des atouts important du développement avec Arduino : une fois que votre carte Arduino est programmée pour utiliser le port série, vous pouvez tester cette communication simplement depuis le Terminal Série du logiciel Arduino. Vous êtes sûr de cette façon que çà fonctionne, vous pouvez vérifier manuellement les réponses d'Arduino à différentes chaînes, etc...

Une fois fait, on peut lancer le programme Processing qui va communiquer avec Arduino en étant sûr de la réponse d'Arduino. Cette procédure est essentielle pour la mise au point des communications entre Processing et Arduino.

*Le test côté Processing*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

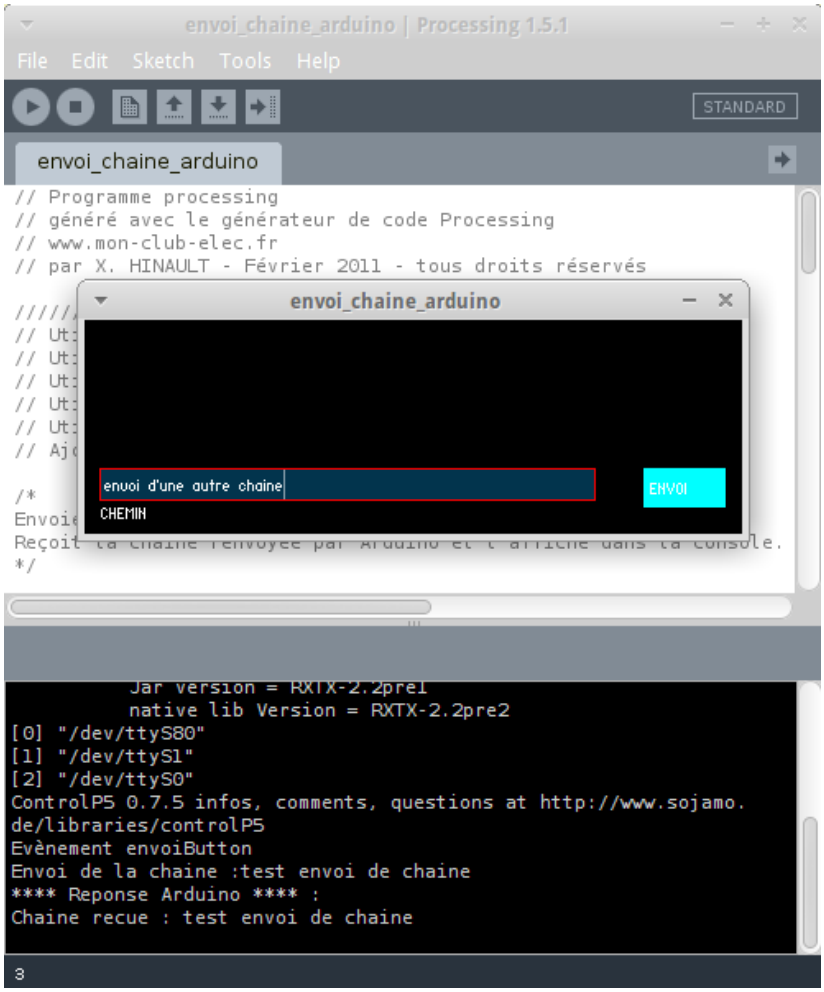
```
// Programme processing
// généré avec le générateur de code Processing
// www.mon-club-elec.fr
// par X. HINAULT - Février 2011 - tous droits réservés

/////////////// Description du programme ////////////
// Utilise un/des objets String (chaîne de caractère)
// Utilise le port Serie
// Utilise la librairie GUI controlP5
// Utilise un/des bouton(s) simple(s) (Button)
// Utilise un/des champ(s) texte (Textfield)
// Ajoute un bouton et un champ pour chemin fichier

/*
Envoie vers Arduino une chaîne saisie dans un champ texte.
Reçoit la chaine renvoyée par Arduino et l'affiche dans la console.
*/

// XXXXXXXXXXXXXXXXXXXXX ENTETE DECLARATIVE XXXXXXXXXXXXXXXXXXXXX

// inclusion des librairies utilisées

import processing.serial.*; // importe la librairie série processing

import controlP5.*; // importe la librairie GUI controlP5
// cette librairie doit être présente dans le répertoire /libraries du répertoire Processing
// voir ici : http://www.sojamo.de/libraries/controlP5/

// déclaration objets

String str1 = "machaine"; // déclare un objet String (chaine de caractère)
String chaineEnvoi = ""; // déclare un objet String (chaine de caractère)

// --- port Série ---
Serial  myPort; // Création objet désignant le port série

ControlP5 controlP5; // déclare un objet principal de la librairie GUI controlP5


Button envoiButton; // déclare objet Button

Textfield chaineText; // déclare des objets Textfield

// déclaration variables globales

//------ déclaration des variables de couleur utiles ----
int jaune=color(255,255,0);
int vert=color(0,255,0);
int rouge=color(255,0,0);
int bleu=color(0,0,255);
int noir=color(0,0,0);
int blanc=color(255,255,255);
int bleuclair=color(0,255,255);
int violet=color(255,0,255);


// XXXXXXXXXXXXXXXXXXXXX  Fonction SETUP XXXXXXXXXXXXXXXXXXXXX

void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

        // ---- initialisation paramètres graphiques utilisés
        colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
        fill(0,0,255); // couleur remplissage RGB
        stroke (0,0,0); // couleur pourtour RGB
        rectMode(CORNER); // origine rectangle : CORNER = coin sup gauche | CENTER : centre
        imageMode(CORNER); // origine image : CORNER = coin sup gauche | CENTER : centre
        //strokeWeight(0); // largeur pourtour
        frameRate(30);// Images par seconde

        // --- initialisation fenêtre de base ---
        size(400, 130); // ouvre une fenêtre xpixels  x ypixels
```

```
        background(0,0,0); // couleur fond fenetre

// --- initialisation des objets et fonctionnalités utilisées ---

        //------------- initialisation port série ----
        println(Serial.list()); // affiche dans la console la liste des ports séries
        // Vérifier que le numéro du port série utilisé est le meme que celui utilisé avec
Serial.list()[index]
        myPort = new Serial(this, Serial.list()[0], 115200); // Initialise une nouvelle instance du
port Série
        //myPort = new Serial(this, "/dev/ttyUSB0", 115200); // Initialise une nouvelle instance du
port Série
        myPort.bufferUntil('\n'); // attendre arrivée d'un saut de ligne pour générer évènement
série

        //======== Initialisation Objets GUI ControlP5 =========

        controlP5 = new ControlP5(this); // initialise l'objet principal de la librairie GUI
controlP5

        // typeObjet nomObjet=controlP5.addObjet(paramètres); // pour info : déclaration /
initialisation possible en 1 ligne
        // Textfield field = controlP5.addTextfield("myWindowTextfield",70,130,100,20); // exemple

        //======== Initialisation Objets Button =========

        //---- le bouton envoi chaine
        envoiButton=controlP5.addButton("envoiButton",0,width-60,height-40,50,24); // initialise et
ajoute un Button au ControlP5
        envoiButton.setLabelVisible(true); // affichage des labels
        envoiButton.setLabel("ENVOI"); // fixe label du bouton
        envoiButton.setColorActive(color(255,0,0)); // fixe la couleur active
        envoiButton.setColorForeground(color(0,255,255)); // fixe couleur avant

        //======== Initialisation Objets Textfield =========

        //---- champ texte saisie chaine
        chaineText=controlP5.addTextfield("cheminText",10,height-40,300,20); // initialise et
ajoute un Textfield au ControlP5
        chaineText.setAutoClear(false); // autoeffacement après return
        chaineText.setValue(chaineEnvoi); // initialise Texte du champ
        chaineText.setLabelVisible(true); // affichage des labels
        chaineText.setLabel("CHEMIN"); // fixe label
        chaineText.setColorActive(color(255,0,0)); // fixe la couleur active
        chaineText.setColorForeground(color(0,255,255)); // fixe couleur avant



} // fin fonction Setup

// XXXXXXXXXXXXXXXXXXXXX Fonction Draw XXXXXXXXXXXXXXXXXXX

void  draw() { // fonction exécutée en boucle

        //------------- Code pour port série ----
        //  myPort.write('H'); // envoie le caractère sur le port Série
        // myPort.write("chaine\n"); // envoie la chaine suivi saut ligne sur le port Série
        // Accès à la valeur des objets GUI controlP5

        //--- accès à la valeur courante du Button ---
        //println("Valeur Button= "+ b1.value());
        //delay(100);

        //--- accès à la valeur courante du Textfield ---
        //println("Valeur Textfield= "+ tf1.value());
        //println("Texte Textfield= "+ tf1.getText());
        //delay(100);



        // while(true); // stoppe boucle draw

} // fin de la fonction draw()

// XXXXXXXXXXXXXXXXXXXXX Autres Fonctions XXXXXXXXXXXXXXXXXXXXX
```

```
        //------------- Fonction de gestion des évènements série ----
         void serialEvent (Serial myPort) { // fonction appelée lors de la survenue d'un évènement
série

        // ******** Gestion de la valeur reçue sur le port série : **********

        String inString = myPort.readStringUntil('\n'); // chaine stockant la chaîne reçue sur le
port Série
        // saut de ligne en marque de fin

        if (inString != null) { // si la chaine recue n'est pas vide


                print (inString); // affichage brut de la chaine recue


        } // fin condition chaine recue pas vide


} // fin de la fonction de gestion des évènements Série

// Gestion des évènements des objets GUI controlP5 ----

//------ fonction gestion globale des évènements GUI controlP5
public void controlEvent(ControlEvent theEvent) {
        //println(theEvent.controller().name());// affiche le nom de l'évènement
}

        //---- evenement bouton envoi chaine

void envoiButton(int theValue) { // fonction évènement Button de meme nom - reçoit la valeur

        println("Evènement envoiButton");

        myPort.write(chaineText.getText()+"\n"); // envoie la chaine suivi saut ligne sur le port
Série
        print("Envoi de la chaine :");
        print(chaineText.getText()+"\n");
        chaineText.setValue(""); // vide le champ texte


} // fin evènement bouton envoi

// ------ gestion évènement Textfield ------

        //---- evenement champ texte chemin fichier

public void chaineText(String theText) { // fonction évènement Textfield de meme nom - déclenché
par return - reçoit la chaine

        //println("Evènement CheminText avec valeur = "+theText);

        chaineEnvoi=theText; // mémorise le contenu du champ

        //println("Le chemin est :"+chaineEnvoi);

} // fin evènement champ texte chemin fichier


//XXXXXXXXXXXXXXXXXX Fin du programme XXXXXXXXXXXXXXXXX
```

Avant de lancer l'exécution, aller dans Tools > CreateFont pour ajouter la police DejaVuSans-9 qui est utilisée par ce programme.

Ensuite, lancer l'exécution du programme : un champ de saisie apparaît. Saisir une chaîne et cliquer sur le bouton envoi. La réponse d'Arduino doit s'afficher dans la console.



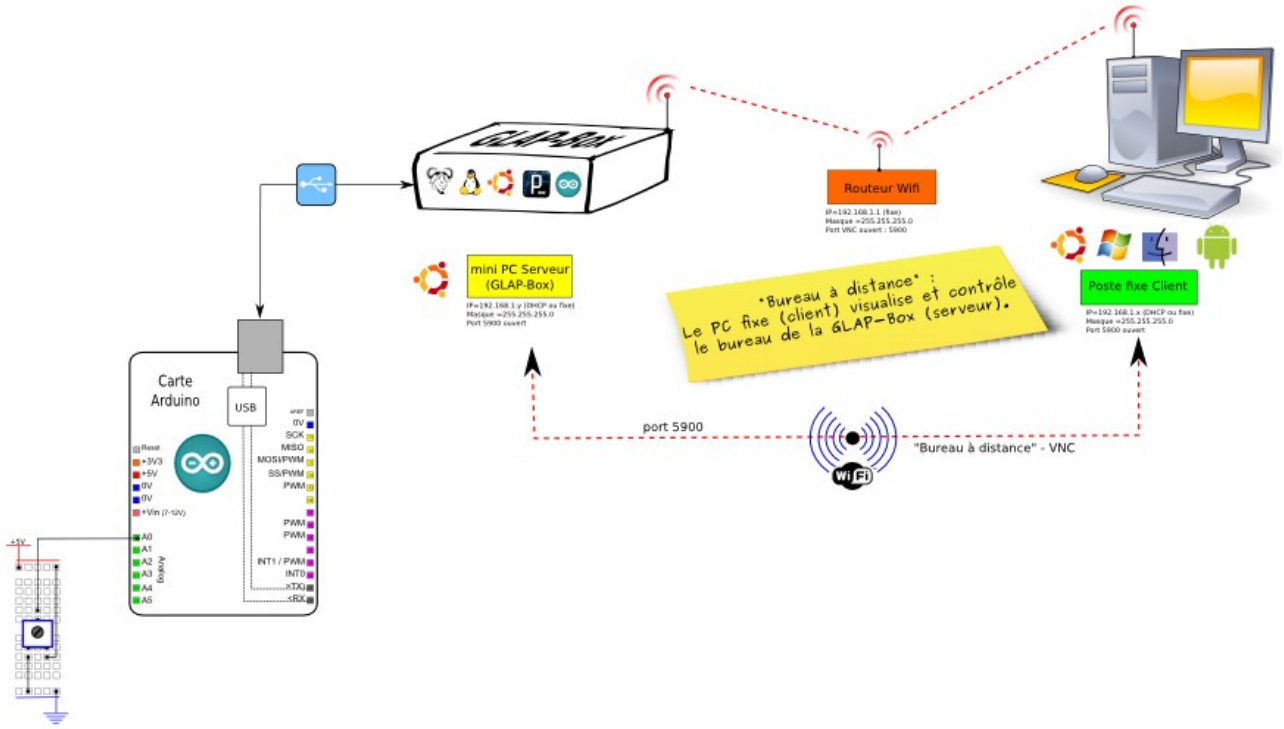

C'est tout bête, mais au moins vous êtes sûr que la communication USB entre Arduino et Processing se fait bien ! Ce qui suppose que le port USB soit bien reconnu notamment... Sur la GLAP-Box, çà fonctionne normalement tout de suite par simple/copier coller du code ci-dessus !

Remarquer également tout l'intérêt d'une GLAP-Box embarquée : vous développez votre code sur votre poste fixe aussi bien pour Arduino que Processing que les 2 (sous réserve de disposer des logiciels Arduino et Processing bien sûr) . Une fois fait, un simple copier/coller vous permet de l'exécuter à distance sur la GLAP-Box. De la même façon, tous les codes existants en exécution fixe sont réutilisables en embarqué !

### Test interface graphique oscilloscope

*Matériel*

Ce test nécessite une carte Arduino connectée à la GLAP-Box.




Idéalement, connecter également un capteur ou une résistance variable sur la broche analogique A0 de la carte Arduino.

*Explications*

La carte Arduino va réaliser une mesure sur la voie analogique A0 et la valeur sera envoyée vers Processing qui affichera le résultat sous forme graphique.

Le code Arduino réalise la mesure et envoie une chaîne sur le port Série.

Le code Processing va consister à afficher sous forme graphique la valeur reçue sur le port série.

*Le test côté Arduino*

Ouvrir Arduino sur la GLAP-Box

Copier/coller le code suivant :

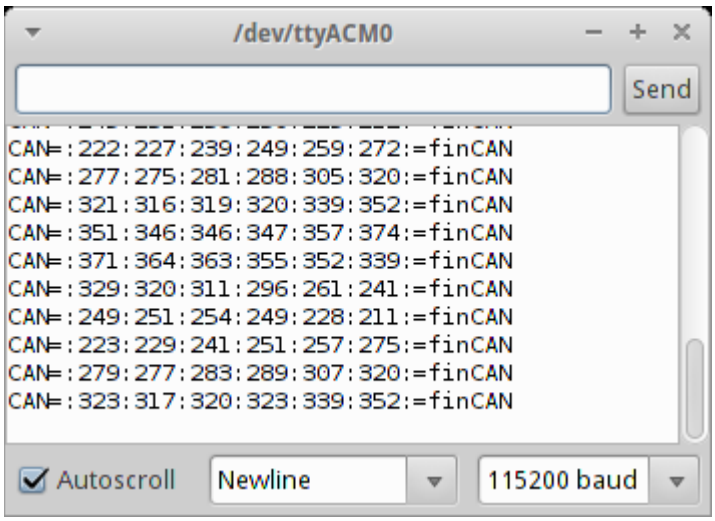
```
// --- Programme Arduino ---
// can_Graph_pc_v3
// par X. HINAULT - 03/2011
// wwww.mon-club-elec.fr

// --- Que fait ce programme ? ---
/* Envoie la valeur brute de la conversion analogique sur 6 voies, sur le port série
sous la forme "CAN=:val0:val1:val2:val3:val4:val5:=finCAN"
*/

// --- Fonctionnalités utilisées ---
// Utilise la connexion série vers le PC
// Utilise la conversion analogique numérique 10bits sur les 6 voies analogiques

// --- Circuit à réaliser ---
// Connexion série entre la carte Arduino et le PC (utilise les broches 0 et 1)

// Broches analogiques : connecter les tension à mesurer sur les broches

//*************** Entête déclarative *******
// A ce niveau sont déclarées les librairies, les constantes, les variables...

// --- Inclusion des librairies utilisées ---
```

```
// --- Déclaration des constantes ---

// --- constantes des broches ---

const int Voie[6]={0,1,2,3,4,5}; //declaration constante de broche analogique

// --- Déclaration des variables globales ---
int mesure_brute=0;// Variable pour acquisition résultat brut de conversion analogique numérique


// --- Initialisation des fonctionnalités utilisées ---

//*************** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et 1 seule fois, au démarrage du programme

void setup()   { // debut de la fonction setup()

// --- ici instructions à exécuter au démarrage ---

Serial.begin(115200); // initialise connexion série à 115200 bauds
// IMPORTANT : régler le terminal côté PC avec la même valeur de transmission




} // fin de la fonction setup()
// ******************************************************************************

//*************** FONCTION LOOP = Boucle sans fin = coeur du programme *************
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino est sous tension

void loop(){ // debut de la fonction loop()

// --- ici instructions à exécuter par le programme principal ---

//---- envoie sur le port série les 6 mesures sous la forme CAN=:val0:val1:val2:val3:val4:val5:=fin
CAN
//--- les : sont utilisés comme séparateur par le programme Processing cté PC

Serial.print("CAN=:"); // chaine de début

for (int i=0; i<6; i++) { // défile les vois analogiques

  // acquisition conversion analogique-numerique (CAN) sur la voie analogique
  mesure_brute=analogRead(Voie[i]);

  // affiche valeur numerique entière ou à virgule au format décimal
  Serial.print(mesure_brute);
  Serial.print(":"); // séparateur entre les valeurs

} // fin for i

  Serial.println("=finCAN"); // fin de la chaine + saut de ligne

  delay(100); // fixe une pause entre 2 mesures
  // plus le délai est court, plus la trace est rapide

} // fin de la fonction loop() - le programme recommence au début de la fonction loop sans fin
// ******************************************************************************

// --- Fin programme ---
```

Programmer la carte Arduino.

Une fois la carte Arduino programmée, pour vérifier le bon fonctionnement du programme :

- ouvrir le Terminal Série, fixer le débit à 115200
- vous devez voir défiler une série de chaine correspondant aux mesures sur les 6 voies analogiques simultanément :




- Une fois que vous êtes sûr que çà marche bien, **fermer la fenêtre Terminal.** Ne pas oublier de le faire ! Sinon, le port série de la carte Arduino est déjà occupé et Processing ne pourra pas s'y connecter...

Même remarque que précédemment. Vous êtes sûr de cette façon que çà fonctionne et que Arduino envoie bien sur le Terminal Série les chaînes attendues.

Une fois fait, on peut lancer le programme Processing qui va communiquer avec Arduino en étant sûr du bon fonctionnement d'Arduino. Cette procédure est essentielle pour la mise au point des communications entre Processing et Arduino.

*Le test côté Processing*

Ouvrir Processing sur la GLAP-Box

Copier/coller le code suivant :

```
// Programme processing
// généré avec le générateur de code Processing
// X. HINAULT - Mars 2011 - tous droits réservés
// www.mon-club-elec.fr

// ------- Licence du code de ce programme -----
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License,
//  or any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

/////////////// Description du programme ////////////
// Utilise le port Serie
// Utilise une police texte

/////////////// ENTETE DECLARATIVE ////////////

// inclusion des librairies utilisées

import processing.serial.*; // importe la librairie série processing

import controlP5.*; // importe la librairie GUI controlP5
// cette librairie doit être présente dans le répertoire /libraries du répertoire
```

```
Processing
// voir ici : http://www.sojamo.de/libraries/controlP5/

// déclaration objets

// --- port Série ---
Serial  myPort; // Création objet désignant le port série

ControlP5 controlP5; // déclare un objet principal de la librairie GUI controlP5

// --- police texte ---
PFont fontA; // crée un objet de police texte

// déclaration variables globales

//------ déclaration des variables de couleur utiles ----
int jaune=color(255,255,0);
int vert=color(0,255,0);
int rouge=color(255,0,0);
int bleu=color(0,0,255);
int noir=color(0,0,0);
int blanc=color(255,255,255);
int bleuclair=color(0,255,255);
int violet=color(255,0,255);

//--- variables de configuration générales des tracés
int nombreVoies=1; // nombre de voies de mesure
boolean niveauxLogiques=false; // pour affichage niveaux logiques sur tracé
int heightTrace, widthTrace; // variable taille tracé
int decalageGauche=100;
int taillePoliceTrace=9;

// décla tableau  : int[] numbers = new int[3];
// ou : int[] numbers = { 90, 150, 30 };

 int[] xRef = new int [nombreVoies];         // tableau variable x de référence (coin
inf gauche du tracé)
 int[] yRef= new int[nombreVoies];       // tableau variable y de référence (coin inf
gauche du tracé)

 int[] couleurVoie= new int [6]; // tableau de variables de couleur du tracé des voies
analogiques

 int[] xPos = new int [nombreVoies];         // tableau variable abscisse  - x
 int[] xPos0= new int[nombreVoies];       // tableau variable mémorisation xPos n-1

float[] yPos=new float [nombreVoies];  // tableau variable yPos - ordonnée
float[] yPos0=new float[nombreVoies]; // tableau variable yPos n-1

 int[] xTextPos= new int[nombreVoies]; // tableau variable position texte dans la
fenêtre
 int[] yTextPos=new int[nombreVoies]; // tableau variable position texte dans la fenetre

 float coeff=1.0; // coeff mutiplicateur
 float base=0.0; // base du tracé en Volt
 boolean etatPause=false; // variable témoin appui sur bouton stop

// déclaration objets controlP5

Toggle togglePause; // déclare un/des objets Toggle

Slider sliderCoeff; // déclare un/ des objet Slider
Slider sliderBase; // déclare un/ des objet Slider

ControlWindow cw1; // déclare un/des objet fenetre ControlWindow

/////////////// Fonction SETUP ////////////
```

```
void setup(){ // fonction d'initialisation exécutée 1 fois au démarrage

    // ---- initialisation paramètres graphiques utilisés
    colorMode(RGB, 255,255,255); // fixe format couleur R G B pour fill, stroke, etc...
    fill(0,0,255); // couleur remplissage RGB
    stroke (0,255,0); // couleur pourtour RGB
        strokeWeight(1); // largeur du trait en pixels
    rectMode(CORNER); // origine rectangle coin sup gauche

    // --- initialisation fenêtre de base ---
    //size(200, 200); // ouvre une fenêtre xpixels  x ypixels
        size(int(screen.width*0.4), int(screen.height*0.8));
        // 90% de l'écran par défaut - régler au besoin -
        // viser taille maxi en hauteur - taille fenêtre (x,y)

    background(0,0,0); // fond fenetre en noir

        //----- initialise variables utilisées pour les graphiques

        // tracé en largeur pleine
        heightTrace=(height-20)/nombreVoies;
        widthTrace=width-decalageGauche;


        // variables indexées
        for (int i=0; i<nombreVoies; i++) { // passe en revue les éléments de tableau
pour n voies

          xPos[i] =1;         // variable abscisse  - x
          xPos0[i]=1;       // variable mémorisation xPos n-1

          yPos[i]=1;  // variable yPos - ordonnée
          yPos0[i]=1; // variable yPos n-1

          xTextPos[i]=0; // variable position texte dans la fenêtre
          yTextPos[i]=50;

           //---- point sup gauche du tracé ---
          xRef[i]=decalageGauche;
          yRef[i]=heightTrace*i;


        }

        //--- les couleurs des voies ----
        couleurVoie[0]=rouge;
        couleurVoie[1]=violet;
        couleurVoie[2]=jaune;
        couleurVoie[3]=bleuclair;
        couleurVoie[4]=blanc;
        couleurVoie[5]=vert;

// --- initialisation des fonctionnalités utilisées ---

    //------------- initialisation de la police texte - à mettre avant série ----
    fontA = loadFont("DejaVuSans-9.vlw"); // charge le fichier police dans l'objet
police texte
    // ce fichier doit être présent dans un rép <data> dans le répertoire du programme
    // pour générer un fichier de police à partir des polices système aller dans Tools >
create Fonts
    // voir également http://processing.org/learning/text/

    textFont(fontA, 9); // Initialise la police et sa taille (en pixels)

    //------------- initialisation port série ----
    println(Serial.list()); // affiche dans la console la liste des ports séries
    myPort = new Serial(this, Serial.list()[0], 115200); // Initialise une nouvelle
instance du port Série
```

```
        //myPort = new Serial(this, "/dev/ttyACM0", 115200); // Initialise une nouvelle
instance du port Série
    myPort.bufferUntil('\n'); // attendre arrivée d'un saut de ligne pour générer
évènement série


//------- tracé initial ---------

        traceInitial();


//======== Initialisation Objets GUI ControlP5 =========

    controlP5 = new ControlP5(this); // initialise l'objet principal de la librairie GUI
controlP5

    // typeObjet nomObjet=controlP5.addObjet(paramètres); // pour info : déclaration /
initialisation possible en 1 ligne
    // Textfield field = controlP5.addTextfield("myWindowTextfield",70,130,100,20); //
exemple

    //======== Initialisation Objets ControlWindow =========

    // addControlWindow(String theWindowName,int theX, int theY, int theWidth, int
theHeight,int theFrameRate)
    cw1=controlP5.addControlWindow("fenetre",50,50,250,150);// ajoute une fenetre au
ControlP5

    // méthodes propres à l'objet ControlWindow
    cw1.hideCoordinates(); //masque les coordonnées
    cw1.setBackground(color(0,0,0));
    cw1.frameRate(15); // fixe le nombre de rafraichissement par seconde de la fenetre
    //cw1.setColorActive(color(255,0,0)); // fixe la couleur active
    cw1.setTitle("Controle Oscillo"); // titre de la fenetre
    //cw1.setLocation(100, 100) ; // fixe la localisation dans la fenetre ?
    //cw1..setUndecorated(true); // fixe la bordure de la fenêtre

    // ajout de controles à la fenetre ControlWindow
    // nomObjet.setWindow(cw1); // met l'objet dans la fenêtre
    // b1.setWindow(cw1); // met l'objet dans la fenêtre
    // t1.setWindow(cw1); // met l'objet dans la fenêtre
    // s1.setWindow(cw1); // met l'objet dans la fenêtre

    //======== Initialisation Objets Toggle =========

    // addToggle(String theName, boolean theDefaultValue, float theX, float theY, int
theWidth, int theHeight)
    togglePause=controlP5.addToggle("togglePause",false,10,90,20,20); // initialise et
ajoute un Button au ControlP5

    // méthodes propres à l'objet Toggle
    togglePause.setMode(ControlP5.DEFAULT); // fixe le mode de fonctionnement du
Toggle : ControlP5.DEFAULT ou ControlP5.SWITCH
        togglePause.setWindow(cw1); // met l'objet dans la fenêtre

    //t2.setMode(ControlP5.SWITCH); // fixe le mode de fonctionnement du Toggle :
ControlP5.DEFAULT ou ControlP5.SWITCH

    // méthodes communes à tous les controles (objet Controller)
    togglePause.setLabelVisible(true); // affichage des labels
    togglePause.setLabel("Pause"); // fixe label objet
    togglePause.setDecimalPrecision(2); // fixe la précision
    togglePause.setColorActive(color(255,0,0)); // fixe la couleur active
    togglePause.setColorBackground(color(255,0,255)); // fixe couleur fond
    togglePause.setColorForeground(color(0,0,255)); // fixe couleur avant
    togglePause.setColorCaptionLabel(color(0,255,255)); // fixe couleur Label
    togglePause.setColorValueLabel(color(0,0,255)); // fixe la couleur valeur
```

```
    // setImages(PImage theImageDefault,PImage theImageOver, PImage
theImageActive,PImage theImageHighlight)
    // les images doivent etre de la meme taille que bouton, dans rép prog, type .jpg
.png ..
    // un toggle n'utilise que image Default et Active
    //t1.setImages(loadImage("imageDefault.png"),loadImage("imageDefault.png"),
loadImage("imageActive.png"),loadImage("imageDefault.png"));


    //======== Initialisation Objets Sliders =========

    // addSlider(theName, theMin, theMax, theDefaultValue, theX, theY, theW, theH)
    sliderCoeff=controlP5.addSlider("sliderCoeff",1,10,1,10,10,100,15); // ajoute un
Slider au ControlP5
        sliderCoeff.setWindow(cw1); // met l'objet dans la fenêtre
    //s1 = (Slider)controlP5.controller("MonSlider1"); // initialise l'objet Slider
déclaré

    // méthodes propres à l'objet Slider
    sliderCoeff.setNumberOfTickMarks(10); // fixe le nombre crans - n+1 pour n valeurs
    //sliderCoeff.setNumberOfTickMarks((int(s1.max())+1); // fixe le nombre crans - n+1
pour n valeurs
    sliderCoeff.showTickMarks(true); // affichage des repères
    sliderCoeff.setSliderMode(Slider.FLEXIBLE); // fonctionnement du slider FLEXIBLE ou
FIX

    // méthodes communes à tous les controles (objet Controller)
    sliderCoeff.setLabelVisible(true); // affichage des labels
    sliderCoeff.setLabel("Coefficient"); // fixe label objet
    sliderCoeff.setDecimalPrecision(2); // fixe la précision
    sliderCoeff.setColorActive(color(255,0,0)); // fixe la couleur active
    sliderCoeff.setColorBackground(color(255,255,0)); // fixe couleur fond
    sliderCoeff.setColorForeground(color(0,0,255)); // fixe couleur avant
    //s1.setArrayValue(new float[] {100,255} ); // fixe les valeurs min/max du Slider ?
    sliderCoeff.setColorCaptionLabel(color(0,255,255)); // fixe couleur Label
    sliderCoeff.setColorValueLabel(color(0,0,255)); // fixe la couleur valeur

    //======== Initialisation Objets Sliders =========

    // addSlider(theName, theMin, theMax, theDefaultValue, theX, theY, theW, theH)
    sliderBase=controlP5.addSlider("sliderBase",0,4.5,0,10,50,100,15); // ajoute un
Slider au ControlP5
        sliderBase.setWindow(cw1); // met l'objet dans la fenêtre
    //s1 = (Slider)controlP5.controller("MonSlider1"); // initialise l'objet Slider
déclaré

    // méthodes propres à l'objet Slider
    sliderBase.setNumberOfTickMarks(10); // fixe le nombre crans - n+1 pour n valeurs
    //sliderCoeff.setNumberOfTickMarks((int(s1.max())+1); // fixe le nombre crans - n+1
pour n valeurs
    sliderBase.showTickMarks(true); // affichage des repères
    sliderBase.setSliderMode(Slider.FLEXIBLE); // fonctionnement du slider FLEXIBLE ou
FIX

    // méthodes communes à tous les controles (objet Controller)
    sliderBase.setLabelVisible(true); // affichage des labels
    sliderBase.setLabel("Base"); // fixe label objet
    sliderBase.setDecimalPrecision(2); // fixe la précision
    sliderBase.setColorActive(color(255,0,0)); // fixe la couleur active
    sliderBase.setColorBackground(color(255,255,0)); // fixe couleur fond
    sliderBase.setColorForeground(color(0,0,255)); // fixe couleur avant
    //s1.setArrayValue(new float[] {100,255} ); // fixe les valeurs min/max du Slider ?
    sliderBase.setColorCaptionLabel(color(0,255,255)); // fixe couleur Label
    sliderBase.setColorValueLabel(color(0,0,255)); // fixe la couleur valeur
```

```
    } // fin fonction Setup

/////////////// Fonction Draw ////////////

void  draw() { // fonction exécutée en boucle


    // while(true); // stoppe boucle draw

} // fin de la fonction draw()

/////////////// Autres Fonctions ////////////

    //------------- Fonction de gestion des évènements série ----
     void serialEvent (Serial myPort) { // fonction appelée lors de la survenue d'un
évènement série

    // ******** Gestion de la valeur reçue sur le port série : **********

    String inString = myPort.readStringUntil('\n'); // chaine stockant la chaîne reçue
sur le port Série
    // saut de ligne en marque de fin


    if ((inString != null) && (etatPause==false)) { // si la chaine recue n'est pas vide
et toggle pause false


        inString = trim(inString); // enlève espace blancs de la chaine recue

                println("Chaine reçue="+inString); // debug

                //---- extraction des valeurs à partir de la chaine reçue ----
                // la chaine reçue avec la valeur des 6 mesures est sous la forme
CAN=:val0:val1:val2:val3:val4:val5:=finCAN

                String[] inSubstring = split(inString, ':'); // extrait

                    println(inSubstring[0]); // debug

              if (match(inSubstring[0],"CAN=")!=null) { // pour prise en compte
uniquement des chaines débutant par CAN:

                //--- déclaration tableaux utiles
                int[] inByte_brut= new int[6]; // tableau pour valeur reçue en valeur
numérique entiere
                float[] inByte= new float[6]; // tableau pour valeur reçue en valeur
numérique décimale

                //---- extraction des valeurs de CAN à partir de la chaine
                for (int i=0; i<nombreVoies; i++) {

                     print("Valeur CAN("+i+")= "+ inSubstring[i+1]); // debug

                     inByte_brut[i]=int(inSubstring[i+1]); // conversion valeur reçue en
valeur numérique entiere
                     print( " soit en valeur entière : " + inByte_brut[i]); // debug

                     inByte[i]=float(inSubstring[i+1]); // conversion valeur reçue en
valeur numérique décimale
                     print( " soit en valeur décimale : " + inByte[i]); // debug

                     println("");
                }

                  println(inSubstring[7]); // debug - chaine de fin de chaine

           if ((match(inSubstring[0],"CAN=")!=null) && (match(inSubstring[7],"=finCAN")!
```

```
=null)) { // si la chaine reçue est valide

                  println("La chaine CAN reçue est valide ! "); // debug

          //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx tracé des courbes
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                 for (int nbTrace=0; nbTrace<nombreVoies; nbTrace++) { // trace toutes
les voies nécessaires

                    inByte[nbTrace]=inByte[nbTrace]-(base*1023.0/5.0); // adaptation de
la base du tracé

                    // ré-échelonne la valeur pour affichage
                     inByte[nbTrace] = map(inByte[nbTrace], 0, 1023, 0, heightTrace); //
adaptation valeur CAN à l'ordonnée tracé

                     yPos[nbTrace]=inByte[nbTrace]; // l'ordonnée est la valeur reçue
par le port série

                     yPos[nbTrace]=yPos[nbTrace]*coeff; // adaptation coeff

                    //********** affichage numérique de la valeur reçue et autres
calculs *****

                     //---- calcul de la tension à afficher --------
                     float tension=float(inByte_brut[nbTrace]);
                     tension=tension*5000;
                     tension=tension/1023;
                     tension=tension; // adaptation tracé

                    // dessin d'un rectangle sous le texte avant affichage pour effacer
texte précédent
                     fill(0,0,0); // couleur de remplissage idem fond
                     stroke( couleurVoie[nbTrace]); // fixe la couleur utilisée pour le
tracé en RVB
                     strokeWeight(1); // largeur du trait en pixels
                     rect(xTextPos[nbTrace],  yRef[nbTrace]+yTextPos[nbTrace]-
taillePoliceTrace-2 , 75, taillePoliceTrace+5);

                      // Use fill() to change the value or color of the text
                      fill(couleurVoie[nbTrace]); // couleur pour la police
                      text("         ", xTextPos[nbTrace]+2, yRef[nbTrace]
+yTextPos[nbTrace]); //efface la valeur précédente
                      //text(inByte_brut, xTextPos, yTextPos);
                      textFont(fontA, 9);
                      text(tension+" mV", xTextPos[nbTrace]+2, yRef[nbTrace]
+yTextPos[nbTrace]);


                      //*********** gestion du graphique *********

                     // trace la ligne
                     stroke( couleurVoie[nbTrace]); // fixe la couleur utilisée pour le
tracé en RVB
                     strokeWeight(1); // largeur du trait en pixels


                     line (xRef[nbTrace]+xPos0[nbTrace],yRef[nbTrace]+heightTrace-
yPos0[nbTrace],xRef[nbTrace]+xPos[nbTrace],yRef[nbTrace]+heightTrace-yPos[nbTrace]); //
trace une ligne en tenant compte valeur reçue

                     xPos0[nbTrace]=xPos[nbTrace]; // mémorisation xPos n-1
                     yPos0[nbTrace]=yPos[nbTrace]; // mémorisation xPos n-1

                     // à la fin de l'écran revient au début
                     if (xPos[nbTrace] >= widthTrace) {
                       xPos[nbTrace] = 0;
```

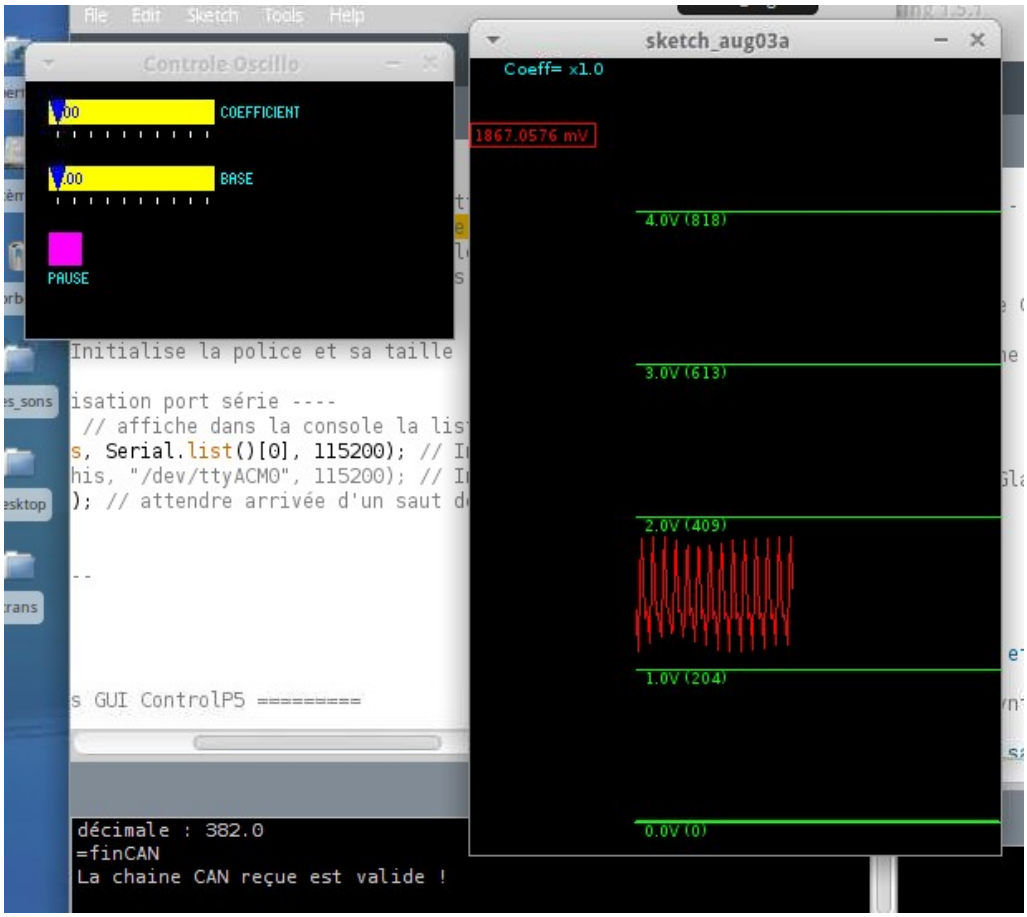
```
                      xPos0[nbTrace]=0; // pour retour de la trace sans ligne

                      background(0,0,0); // couleur du fond


                     //------- tracé initial ---------

                     traceInitial();

                     }  // fin RAZ graphique

                    else {// si pas encore la fin du graphique
                      // incrémente la position horizontale (abscisse)
                      xPos[nbTrace]++;
                     } // fin else


                } // fin for nbTrace

            } // fin si chaine valide

              } // fin prise en compte uniquement des chaines débutant par CAN:

    } // fin condition chaine recue pas vide



} // fin de la fonction de gestion des évènements Série


void traceInitial() {


        for (int nbTrace=0; nbTrace<nombreVoies; nbTrace++) { // boucle pour autant de
tracés que de voies


          for (int i=0; i<5; i++) {// tracé des lignes horizontales

            // --- initialisation avant chaque tracé
            fill(0,255,0); // couleur remplissage RGB
            stroke (0,255,0); // couleur pourtour RGB
            if (i==0) strokeWeight(2); else strokeWeight(1); // largeur du trait en
pixels - 1er trait plus épais

            // trace ligne niveau tension
            line (xRef[nbTrace],yRef[nbTrace]+heightTrace-((heightTrace/5)*i),
xRef[nbTrace] + widthTrace-1,yRef[nbTrace]+heightTrace-((heightTrace/5)*i));

            // affiche valeur tension de la ligne
            textFont(fontA, taillePoliceTrace);
            text(((i/coeff)+base)+"V ("+int(i*1023/5/coeff)+")", xRef[nbTrace]+5,
yRef[nbTrace]+heightTrace-((heightTrace/5)*i)+taillePoliceTrace);

          }

          if (niveauxLogiques==true) { // si dessin niveau logique activé

            stroke (255,255,0); // couleur pourtour RGB
            strokeWeight(2); // largeur du trait en pixels
            fill(255,255,0); // couleur remplissage RGB

            // --- ligne niveau logique bas = 0.3 x Vcc pour carte Arduino (ATmega328)
----
            textFont(fontA, taillePoliceTrace);
            text("Niveau BAS", xRef[nbTrace]+5, (yRef[nbTrace]+heightTrace-
(0.3*coeff*heightTrace)-5));
            line (xRef[nbTrace]+0,(yRef[nbTrace]+heightTrace-(0.3*coeff*heightTrace)),
```

```
xRef[nbTrace]+widthTrace-1,(yRef[nbTrace]+heightTrace-(0.3*coeff*heightTrace)));

            // --- ligne niveau logique haut = 0.6 x Vcc pour carte Arduino (ATmega328)
---
            textFont(fontA, taillePoliceTrace);
            text("Niveau HAUT", xRef[nbTrace]+5, (yRef[nbTrace]+heightTrace-
(0.6*coeff*heightTrace)-5));
            line (xRef[nbTrace]+0,(yRef[nbTrace]+heightTrace-(0.6*coeff*heightTrace)),
xRef[nbTrace]+widthTrace-1,(yRef[nbTrace]+heightTrace-(0.6*coeff*heightTrace)));

          } // fin si dessin niveaux logiques


        } // fin boucle défilement des tracés

        // affichage du coefficient courant
        //text(inByte_brut, xTextPos, yTextPos);
         fill(0,255,255);
         textFont(fontA, 10);
        text("Coeff= x"+coeff,20, 10); // la coordonnée du text est point inf gauche

} // fin tracé initial



// Evènement Toggle
void togglePause(int theValue) { // fonction évènement Toggle de meme nom - reçoit la
valeur
    println("Evènement Toggle Pause avec valeur = "+togglePause.getState());
        etatPause=togglePause.getState(); // met à jour la variable etat pause
}

// ------ gestion évènement Slider ------
void sliderCoeff(float valeur) { // fonction évènement Slider de meme nom - reçoit la
valeur
    println("Evènement Slider Coeff avec valeur = "+valeur);
        coeff=valeur; //MAJ variable coeff

        // affichage du coefficient courant
        //fill(0,0,0);
        //text("                ", 10, 10); //efface la valeur précédente

        // dessin d'un rectangle sous le texte avant affichage pour effacer texte
précédent
        fill(0,0,0); // couleur de remplissage idem fond
        stroke(0,0,0); // fixe la couleur utilisée pour le tracé en RVB
        strokeWeight(0); // largeur du trait en pixels
        rect(10, 0, 75, 10);

        //text(inByte_brut, xTextPos, yTextPos);
         fill(0,255,255);
         textFont(fontA, 10);
        text("Coeff= x"+coeff,20, 10); // la coordonnée du text est point inf gauche

        delay(100); // évite prise en compte multiple

        //traceInitial();
}

// ------ gestion évènement Slider ------
void sliderBase(float valeur) { // fonction évènement Slider de meme nom - reçoit la
valeur
    println("Evènement Slider Base avec valeur = "+valeur);

        base=valeur; // base

        delay(100); // évite prise en compte multiple
}
```

```
/////////////////// fin du programme ////////////////////////////////
```

Avant de lancer l'exécution, aller dans Tools > CreateFont pour ajouter la police DejaVuSans-9 qui est utilisée par ce programme.

Ensuite, lancer l'exécution du programme : un graphique doit apparaître correspondant à l'affichage sous forme oscilloscope de la mesure réalisée par Arduino.




Vous voyez sur votre poste fixe et à distance, en live, les mesures réalisées par votre carte Arduino embarquée à distance : pas mal non ? Ce qu'il faut bien comprendre, c'est qu'à partir du moment où l'accès à la GLAP-Box peut se faire par wifi, votre GLAP-Box peut dès lors se trouver embarquée sur un robot mobile ou une plateforme en hauteur ou extérieure... et vous pouvez visualiser sur le poste fixe les mesures en cours à distance, etc... J'imagine que çà vous donne quelques idées non ?

A noter qu'il suffit de changer le nombre de voies dans le code pour obtenir le même graphique sur 6 voies !

A noter que vous pouvez également stopper l'exécution du code Processing, reprogrammer aussi bien Arduino que Processing et relancer l'exécution SANS TOUCHER PHYSIQUEMENT à la GLAP-Box ! C'est très puissant pour développer vos codes tout en laissant votre montage en situation réelle, à son emplacement de fonctionnement et pas forcément à côté de votre poste fixe.

### *Et aussi : Principe d'utilisation d'un programme Processing au démarrage pour la GLAP-Box*

…

### *Conclusion*

Nous arrivons ainsi au terme du test des fonctions principales disponibles avec la GLAP-Box. A présent, vous allez pouvoir développer toutes sortes d'applications du plus simple au plus évolué, en combinant entre-elles les possibilités embarquées.

## *Tests complémentaires spécifiques utiles pour le GLAP-Bot*

Le GLAP-Bot est un robot mobile propulsé par 2 moteurs CC et embarquant une GLAP-Box. Ici sont décrites quelques procédures utiles spécifiques à ce robot.

## Test des moteurs CC du GLAP-Bot

Ce test permet de vérifier que les moteurs sont bien câblés. **ATTENTION : surélever les roues du robot pour ce test afin d'éviter des mouvements du robot incontrôlés.**

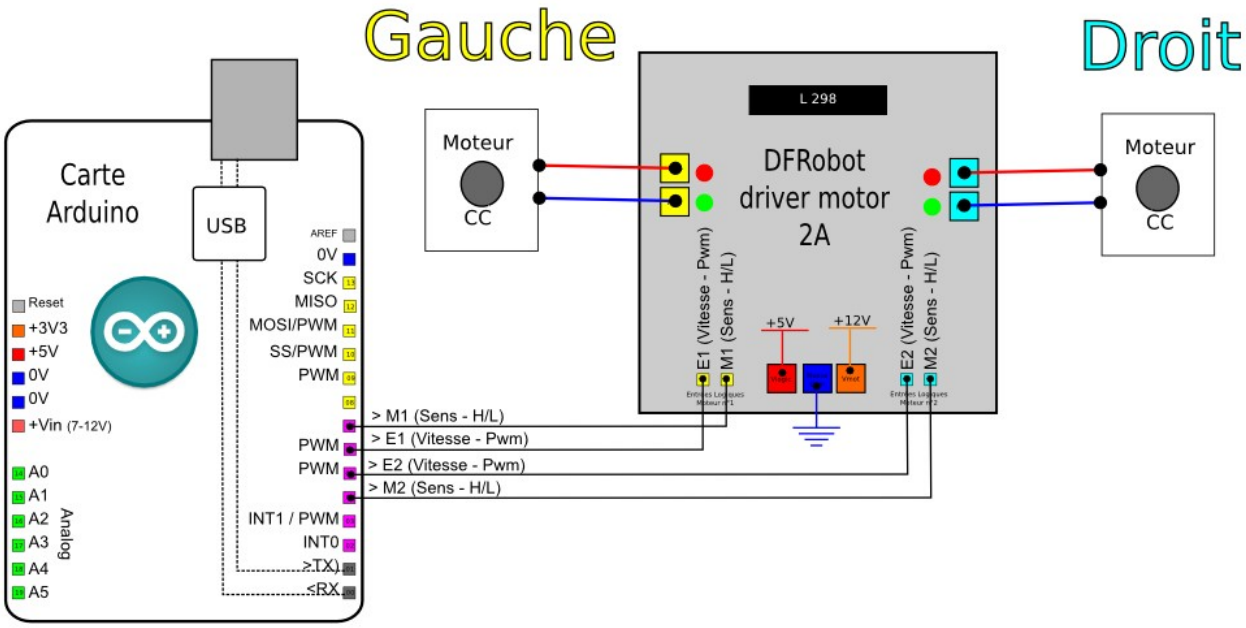
### *Le montage*

Le robot possède 2 moteurs CC droit et gauche qui sont chacun contrôlés par la carte Arduino via une interface de puissance basée sur un L298. Chaque moteur est contrôlé :

- par une broche de sens
- par une broche de vitesse (impulsion PWM)

Sur le GLAP-Bot, on utilise :

- pour le moteur droit :
  - la broche 4 pour la broche de sens
  - la broche 5 (pwm) pour la broche de vitesse
- pour le moteur gauche :
  - la broche 6 (pwm) pour la broche de vitesse
  - la broche 7 pour la broche de sens

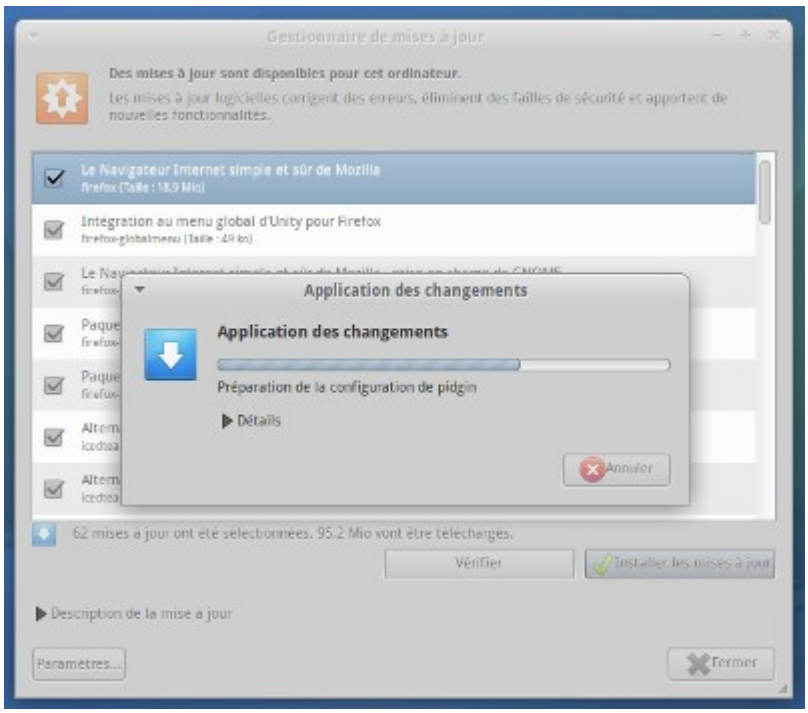



***Le code de test***

***Fonctionnement***

## Mettre à jour la distribution sur la GLAP-Box

On pourra profiter occasionnellement d'une connexion du réseau à internet pour mettre à jour sa GLAP-Box. Ce n'est pas indispensable mais conseillé. A gauche dans le tableau de bord supérieur, cliquer sur l'icône de mise à jour et valider la mise à jour. Laisser le processus se faire tout simplement. Cliquer sur fermer quand c'est terminé.


Gestionnaire de mises à jour
Des mises à jour sont disponibles pour cet ordinateur.
Les mises à jour logicielles corrigent des erreurs, éliminent des failles de sécurité et apportent de nouvelles fonctionnalités.
Le Navigateur Internet simple et sûr de Mozilla
Intégration au menu global d'Unity pour Firefox
Application des changements
Application des changements
Préparation de la configuration de pidgin
Détails
Annuler
62 mises à jour ont été sélectionnées. 95,2 Mio vont être téléchargés.
Vérifier
Installer les mises à jour
Description de la mise à jour
Paramètres...
Fermer

Installation :

```
$ sudo apt-get install libttspico-utils
```

Utilisation :

Truc d'utilisation :

A ce stade, çà ouvre quand même pas mal de possibilités : saisir une longitude, une latitude ou tout autre coefficient à virgule avec un clavier matriciel !

Ici, on utilise le port série, mais tout se passe du côté de l'Arduino et il sera donc possible de réaliser des applications autonomes avec écran LCD par exemple et clavier matriciel pour saisir les valeurs numériques entières ou décimales : à vos idées !

## Annexes techniques

### Les clients VNC sous Xubuntu